Loquendo™ TTS

Multilanguage Text-to-speech Synthesizer

6.5

SDK User's Guide

LoquendoTTS

6.5

SDK User's Guide

Version 6.5.5

21 February 2006

© 2005 Loquendo – All rights reserved

Loquendo confidential

Information in this document is subject to change

No part of this document may be photocopied or reproduced in any form without prior written permission from Loquendo

Loquendo™ is a trademark of Loquendo – Other trademarks are property of their owners

Contents

1 Introduction 5
1.1 Contents 5
1.2 What is Loquendo TTS? 5
2 Text and sentences 7
2.1 Reading modes 7
2.1.1 Multiline, UTF-8 Multiline and UNICODE Multiline Mode 7
2.1.2 Paragraph, UTF-8 Paragraph and UNICODE Paragraph mode 8
2.1.3 XML, UTF-8 XML and UNICODE XML mode 8
2.2 Character sequences (Words) 8
2.2.1 Stress position 8
2.3 Abbreviations and Acronyms 8
2.4 Punctuation marks 9
2.5 Sequences of Digits (Numbers) 9
2.6 Separators 9
3 Working with lexicons 10
3.1 Literal transcriptions 10
3.2 Phonetic transcriptions 11
1.3 Regular expressions 12
3.3.1 Syntax 12
3.3.2 Ambiguities 12
3.3.3 Using regular expressions for find/replace 13
4 Mixed Language Support (optional) 15
5 Control tags 19
5.1 Voice change 20
5.2 Language change 20
5.3 Language guesser configuration 21
5.4 User lexicons 23
5.5 Plugin lexicons 24
5.6 Numbers say as 25
5.7 Phonetic input 27
5.8 Spelling 29
5.9 Read (aloud) punctuation 29
5.10 Read (aloud) control tags 30
5.11 Prosodic pauses 31
5.12 Prominence 32
5.13 Emphasis 33
5.14 Punctuation pause 33
5.15 Speaking rate 34
5.16 Tone (fundamental frequency) 35
5.17 Volume (gain) 36
5.18 Prosody change range 37
5.19 Duration control 39
5.20 Raw signal files playing 40
5.21 Audio mixer capabilities 41
5.22 Bookmarks 49
6 Tools and Samples 50
6.1 Console applications 50
6.2 Web applications 50
6.3 Multi-platform GUI application 50
6.3.1 TTSDirector 51
6.4 Windows only GUI application 53
6.4.1 Edit2Speech 53
6.4.2 LexEditor 56
6.4.3 Eloqwi 60
6.4.4 TTSApp 60

6.4.5 AttsTest....60
6.4.6 TTSDirUpdate....60
7 APPENDIX A: XML support....61
7.1 VOICEXML 1.0: SUPPORTED TAGS AND FORMATS....62
7.2 SSML 1.0 (W3C WD 02 December 2002): SUPPORTED ELEMENTS AND FORMATS....64

1 Introduction

1.1 Contents

The present guide is designed for users and programmers who intend to use the Loquendo™ Text-To-Speech synthesizer in an effective way. This manual is organized in 5 chapters and an appendix:

1. CHAPTER 1: **Introduction** (this chapter, a preliminary description of the Loquendo Text-To-Speech synthesizer)
2. CHAPTER 2: **Text and Sentences** (how to design the input text in order to take advantage of the Loquendo linguistic accuracy in natural language handling)
3. CHAPTER 3: **Working with Lexicons** (how to improve Loquendo™ TTS reading quality by means of exception handling – phonetic transcription and abbreviations)
4. CHAPTER 4: **Control Tags** (how to control and tune the speech quality using synchronous text-embedded commands)
5. APPENDIX A: **XML support** (description of supported XML tags)

Please refer to the "Loquendo™ TTS Programmer's Guide" for any information about the following items:

- Loquendo TTS setup and licensing
- Sample programs shipped with the Loquendo™ TTS SDK
- APIs
- Audio destinations

For every language, please refer to the relative "Loquendo™ Language Reference Guide" (inside the voice CD-ROM distribution) for any information about the following items:

- Language phonemes
- Sequence of Digits (Numbers)
- Plugin lexicons (when available)

1.2 What is Loquendo TTS?

Loquendo™ TTS is a Multilanguage/Multivoice Text-To-Speech synthesizer, peculiar for its very high audio quality and its linguistic accuracy. The Text-To-Speech conversion is a real-time "software-only" process: the number of channels that may be served simultaneously depends on the voice quality and the CPU power.

Loquendo™ TTS is shipped in the form of a library, and all its features are accessed by a set of legacy APIs, that allow the control of every aspect of the TTS process. The speech can be output to a multimedia audio board, a telephone card or a file. In order to use "custom audio destinations" (such as a LAN, or a legacy audio board) the audio destination developer or vendor can provide its own set of callback functions to be interfaced with the Loquendo TTS library (see "Loquendo™ TTS Programmer's Guide" for details).

Loquendo TTS engine is also compliant to Microsoft Speech SDK 4.0 and Microsoft Speech SDK 5.1 (SAPI). All the "required" interfaces are supported, as well as some "optional" ones. This means that any application using the SAPI TTS interfaces is virtually compatible with Loquendo TTS (see "Loquendo™ TTS Programmer's Guide" for the list of SAPI interfaces supported by the present Loquendo TTS release).

The Hardware and Software requirements, as well as the Loquendo™ TTS Setup instructions, including how to obtain a valid license key, are fully described in the "Loquendo™ TTS Programmer's Guide".

2 Text and sentences

This *Guide* describes how Loquendo™ TTS handles the input text. The end user usually does not access the system directly, but through an interface, which may process the text before passing it on to Loquendo™ TTS. Consequently, the operations described below may differ according to the applications using the system. For a more natural voice sound, avoid over-long and complex sentences.

2.1 Reading modes

Nine basic reading modes are possible:

- Multiline (default)
- Paragraph
- XML
- UTF-8 Multiline
- UTF8 Paragraph
- UTF-8 XML
- UNICODE Multiline
- UNICODE Paragraph
- UNICODE XML

Switching from a mode to another can be obtained using appropriate APIs ttsSetInstanceParam (see "Loquendo™ TTS Programmer's Guide") or specifying the appropriate modes as arguments of function ttsRead.

You can test reading modes by using the application Edit2Speech, included with the Loquendo™ TTS SDK. The label UNICODE and UTF-8 specify the format of the input text: UTF-8 is the Unicode Transformation Format that serializes a Unicode code point as sequence of one to four bytes.

2.1.1 Multiline, UTF-8 Multiline and UNICODE Multiline Mode

In the first mode (Multiline), Loquendo™ TTS will ignore single line breaks (\n), considering them as simple formatting characters. Double (or more consecutive) line breaks, very short lines (less than 5 words), and multiple spaces on the same line will generate a single pause.

For instance, consider the following text chunk:

Introduction to the Loquendo™ TTS reading modes

Now we want to describe the "multiline" reading mode of Loquendo TTS, a way in which text can be split in more than a single line.
Thank you
Bye *January 12 2001*

Loquendo TTS will generate a pause after "*Loquendo TTS reading modes"* (double paragraph), after *"Thank you"* (less than 5 words) and after *"Bye"* (multiple spaces), even if there is no punctuation mark. No pause, instead, will be added after "*in which text*".

Multiline is the default reading mode: it is well suited for the most part of documents.

2.1.2 Paragraph, UTF-8 Paragraph and UNICODE Paragraph mode

In this mode each line break will be considered as a paragraph and will produce a pause.

Paragraph is the best mode for reading non-line-terminated texts, such as word processing documents.

2.1.3 XML, UTF-8 XML and UNICODE XML mode

In this mode a non-validating XML parser is used. See APPENDIX A (XML support) for details.

2.2 Character sequences (Words)

A **word** is a sequence of characters delimited by separators (see *Separators,* 2.6). The exact definition of word may depend on the language spoken. For instance, English words are sequences of ASCII characters (included in the range 032-127), while in other European languages, some other ANSI characters (like stressed vowels) are also possible.

In preparing a text, the first rule is to write using the normal rules applying to the grammar. The second rule is to remember that the information you want to convey will be spoken. This means that best results will be achieved if you try to imagine that you are writing a speech or a script, which will then be delivered or "performed" by the TTS.

Only proper names or acronyms should be capitalized or written in uppercase (e.g., "`Il mio amico Gianni lavora in IBM`."). If a text is written entirely in uppercase characters, converting it to lowercase before passing it to Loquendo TTS will usually ensure better results.

2.2.1 Stress position

Loquendo™ TTS automatically assigns the lexical stress to each word.

However, for some languages (Italian, Spanish, German) the automatic stress assignment can be overridden by inserting the stress character after the vowel to be stressed (e.g., "`La fo`rmica del tavolo.`"). In Windows and UNIX systems, accented characters can also be used. Grave and acute accents may correspond to a different pronunciation (e.g. in Italian, *bòtte* and *bótte* are pronounced with an open and a close 'o' respectively).

2.3 Abbreviations and Acronyms

Abbreviations are widely used in written text, especially for the names of government agencies, titles and so on. An abbreviation for a sequence of several words is an acronym, which is generally made up of the initial letters of each of the words.

An abbreviation is pronounced by saying the whole word that the abbreviation stands for (e.g., Sig. => *signor*), whereas an acronym may be spelled out or pronounced as if it were a word (e.g., ACI => *aci*).

Some abbreviations are dealt with automatically; others may be expanded (i.e., associated with the unabbreviated word) by means of the lexicons (see Chapter 3 *Working with Lexicons*).

By default, Loquendo™ TTS spells out sequences consisting entirely of consonants (for example SKF) letter by letter. The "\s" command will make the synthesizer spell out any word (see Chapter 4, *Control Tags*).

If an acronym contains periods, they must not be followed by spaces (e.g., "`S.p.a.`", not "`S. p. a.`"; In this way, the periods in an acronym will be ignored, whereas if the period is followed by a space it is interpreted as a strong terminator, and thus as the end of a sentence.

2.4 Punctuation marks

A separator (like a blank or newline) must follow periods indicating the end of a sentence (e.g., `"Primo enunciato. Secondo."`). Sequences of periods are read as a single period.

The following table summarizes the macroscopic effects produced by punctuation marks and parentheses, for most languages. Note that in Greek language, questions are marked by ";" rather than "?".

Punctuation mark	Description	Effects:
.	Period	*Long pause, conclusive intonation*
...	Dots	*Long pause, suspensive intonation*
!	Exclamation point	*Long pause, conclusive intonation*
?	Question mark	*Long pause, interrogative intonation*
:	Colon	*Pause, conclusive intonation*
;	Semicolon	*Pause, conclusive intonation (except for Greek)*
,	Comma	*Short pause, suspensive intonation*
(	Round bracket	*Short pause, suspensive intonation*
)	Round bracket	*Short pause, suspensive intonation*

Table 1 – Macroscopic effects of punctuation marks

2.5 Sequences of Digits (Numbers)

See the language reference guides.

2.6 Separators

The separators **SPACE, TAB, RETURN, NEWLINE, FORMFEED** are those which are most frequently used for separating words. The strong terminators **colon, semicolon, exclamation point** and **question mark** are also separators. The **period** acts as a separator only when used between digits, whereas the **comma** is always a separator, though its effects will differ according to whether it is used between words or between digits. Other symbols (e.g. the **apostrophe** ‘’’, ‘-‘ or ‘/’) may act as word separators depending on the language. Another separator is the (ASCII 039), providing that it is not a “misspelled” stress character and placed after a vowel.

3 Working with lexicons

Loquendo™ TTS can manage two kinds of language dependent lexicon files for exception handling:

1. The plugin lexicons
2. The user lexicons

Plugin lexicons are provided together with the Language Library for improving the LoquendoTTS capabilities in reading particular kinds of texts (eg. SMS, e-mails) that may present idiosyncratic forms of words, abbreviations, marks, and so on.

The available plugin lexicons can be activated by a specific item of the TTSDirector "Effects" menu (see the relative chapter), or with a control tag inserted in the text, like the following:

\plugin=SMS

To deactivate it, use the following:

\plugin=*SMS

For the list of the available plugin lexicons for a given language, see the relative Language Reference Guide (inside the voice CD-ROM distribution) or the TTSDirector "Effects" menu.

User lexicons are optional (and provided by the user). They should contain user exceptions and transcriptions. A user lexicon file can be setup programmatically by using the appropriate API (ttsNewLexicon - see Loquendo™ TTS Programmer's guide), or directly in the text using appropriate control tags (**\lexicon=<filename>** - see "Control Tags" section).

Several plugin and user lexicons can be loaded on top of each other. The last loaded lexicon will be accessed first, overriding the others in case of conflicting definitions.

The lexicon entries can have three different forms:

1. Literal transcriptions (expansions)
2. Phonetic transcriptions
3. Regular expressions

3.1 Literal transcriptions

Literal transcriptions have the following form:

"word(s)" = "transcription"

They are case insensitive, unless you explicitly require case sensitivity by inserting **\x** at the beginning of the word, as in the following examples:

"\xOK" = "Oklaoma"

"\xok" = "okay"

One or more words can be used on both sides. For instance:

"pio x" = "pio decimo"

“s.p.a”: = “Società per azioni”

“asap” = “as soon as possible”

Although not forbidden, the use of numerical expressions or symbols on the right side of a literal transcription should be avoided, since this would lead to recursions and/or time consuming computations. You should instead use plain words when possible.

3.2 Phonetic transcriptions

Phonetic transcriptions can be added to lexicons, in the following way:

“word(s)” = “\f...”

The expression on the right side is a list of phonetic symbols (separated by hyphens) following the string **\f,** for instance:

”scherzo” = “\fs-k-`E-r-Ts:-o”

See the tables of phonetic symbols, for the available languages, in the specific “Language Reference Guide” included inside every voice distribution.

1.3 Regular expressions

Regular expressions can be used to give more sophisticated rules. The syntax is:

"\rRegular expression" = "Transcription"

The string **\r** informs Loquendo™ TTS that the rule is a regular expression.

For instance[1]:

```
"\r([0-9]+) ?[xX] ?([0-9]+)" = "\1 per \2"
```

3.3.1 Syntax

A regular expression is zero or more branches, separated by '|'. It matches anything that matches one of the branches.
A branch is zero or more pieces, concatenated. It matches a match for the first, followed by a match for the second, etc.
A piece is an atom possibly followed by '*', '+', or '?'. An atom followed by '*' matches a sequence of 0 or more matches of the atom. An atom followed by '+' matches a sequence of 1 or more matches of the atom. An atom followed by '?' matches a match of the atom, or the null string.
An atom is a regular expression in parentheses (matching a match for the regular expression), a range (see below), '.' (matching any single character), '^' (matching the null string at the beginning of the input string), '$' (matching the null string at the end of the input string), a '\' followed by a single character (matching that character), or a single character with no other significance (matching that character).
A range is a sequence of characters enclosed in '[]'. It normally matches any single character from the sequence. If the sequence begins with '^', it matches any single character not from the rest of the sequence. If two characters in the sequence are separated by '-', this is shorthand for the full list of ASCII characters between them (e.g. [0-9]' matches any decimal digit). To include a literal ']' in the sequence, make it the first character (following a possible '^'). To include a literal '-', make it the first or last character.

3.3.2 Ambiguities

If a regular expression could match two different parts of the input string, it will match the one that begins earliest. If both begin in the same place but match different lengths, or match the same length in different ways, life gets messier, as follows.
In general, the possibilities in a list of branches are considered in left-to-right order, the possibilities for '*', '+', and '?' are considered longest-first, nested constructs are considered from the outermost in, and concatenated constructs are considered leftmost-first. The match that will be chosen is the one that uses the earliest possibility in the first choice that has to be made. If there is more than one choice, the next will be made in the same manner (earliest possibility) subject to the decision on the first choice. And so forth.
For example, '(ab|a)b*c' could match 'abc' in one of two ways. The first choice is between 'ab' and 'a'; since 'ab' is earlier, and does lead to a successful overall match, it is chosen. Since the 'b' is already spoken for, the 'b*' must match its last possibility--the empty string--since it must respect the earlier choice.
In the particular case where the regular expression does not use `|' and does not apply `*', `+', or `?' to parenthesized subexpressions, the net effect is that the longest possible match will be chosen. So `ab*', presented with `xabbbby', will match `abbbb'. Note that if `ab*' is tried against `xabyabbbz', it will match `ab' just after `x', due to the begins-earliest rule. (In effect, the decision on where to start the match is the first choice to be made; hence subsequent choices must respect it even if this leads them to less-preferred alternatives.)
After a successful match, you can retrieve a replacement string as an alternative to building up the

[1] This Italian rule means that 12x15 must be read as 12 per 15.

various substrings by hand.
Each character in the source string will be copied to the return value except for the following special characters:

&	The complete matched string (sub-string 0).
\1	Sub-string 1
…	and so on until...
\9	Sub-string 9

3.3.3 Using regular expressions for find/replace[2]

Normally, when you search for a sub-string in a string, the match should be exact. So if we search for a sub-string "abc" then the string being searched should contain these exact letters in the same sequence for a match to be found. We can extend this kind of search to a case insensitive search where the sub-string "abc" will find strings like "Abc", "ABC" etc. That is, the case is ignored but the sequence of the letters should be exactly the same. Sometimes, a case insensitive search is also not enough. For example, if we want to search for numeric digit, then we basically end up searching for each digit independantly. This is where regular expressions come in to our help.
Regular expressions are text patterns that are used for string matching. Regular expressions are strings that contains a mix of plain text and special characters to indicate what kind of matching to do. Here's a very brief turorial on using regular expressions before we move on to the code for handling regular expressions.
Suppose, we are looking for a numeric digit then the regular expression we would search for is "[0-9]". The brackets indicate that the character being compared should match any one of the characters enclosed within the bracket. The dash (-) between 0 and 9 indicates that it is a range from 0 to 9. Therefore, this regular expression will match any character between 0 and 9, that is, any digit. If we want to search for a special character literally we must use a backslash before the special character. For example, the single character regular expression "*" matches a single asterisk. In the table below the special characters are briefly described.

Character	Description
^	Beginning of the string. The expression "^A" will match an ‘A’ only at the beginning of the string.
^	The caret (^) immediately following the left-bracket ([) has a different meaning. It is used to exclude the remaining characters within brackets from matching the target string. The expression "[^0-9]" indicates that the target character should not be a digit.
$	The dollar sign ($) will match the end of the string. The expression "abc$" will match the sub-string "abc" only if it is at the end of the string.
|	The alternation character (|) allows either expression on its side to match the target string. The expression "a|b" will match ‘a’ as well as ‘b’.
.	The dot (.) will match any character.
*	The asterix (*) indicates that the character to the left of the asterix in the expression should match 0 or more times.

[2] This is a brief article by Zafir Anjum which can be useful to understand the use of regular expressions

+	The plus (+) is similar to asterix but there should be at least one match of the character to the left of the + sign in the expression.
?	The question mark (?) matches the character to its left 0 or 1 times.
()	The parenthesis affects the order of pattern evaluation and also serves as a tagged expression that can be used when replacing the matched sub-string with another expression.
[]	Brackets ([and]) enclosing a set of characters indicate that any of the enclosed characters may match the target character.
\{ \}	Quoted braces enclosing a set of characters indicate a matching word

The parenthesis, besides affecting the evaluation order of the regular expression, also serves as tagged expression which is something like a temporary memory. This memory can then be used when we want to replace the found expression with a new expression. The replace expression can specify a & character which means that the & represents the sub-string that was found. So, if the sub-string that matched the regular expression is "abcd", then a replace expression of "xyz&xyz" will change it to "xyzabcdxyz". The replace expression can also be expressed as "xyz\0xyz". The "\0" indicates a tagged expression representing the entire sub-string that was matched. Similarly we can have other tagged expression represented by "\1", "\2" etc. Note that although the tagged expression 0 is always defined, the tagged expression 1,2 etc. are only defined if the regular expression used in the search had enough sets of parenthesis. Here are few examples.

String	Search	Replace	Result
Mr.	(Mr)(\.)	\1s\2	Mrs.
abc	(a)b(c)	&-\1-\2	abc-a-c
bcd	(a|b)c*d	&-\1	bcd-b
abcde	(.*)c(.*)	&-\1-\2	abcde-ab-de
cde	(ab|cd)e	&-\1	cde-cd

4 Mixed Language Support (optional)

If the Mixed Language Support (optional distribution) is installed, the LoquendoTTS includes the latest technologies to approach multilinguality in TTS, such as: the Mixed Language Capability, enabling foreign words to be pronounced correctly without changing the current voice, and the Language Guesser, which makes it possible to identify the different languages in a document, and ensures that automated TTS system will switch language accordingly.

Loquendo TTS approach to mixed-language speech synthesis offers a range of options to face the various situations where texts may occur in different languages or embedding foreign phrases. The most challenging target is to make a monolingual TTS voice read a foreign language text. A Foreign Pronunciation Strategy allows mixing phonetic transcriptions of different languages, relying on a Phoneme Mapping algorithm making foreign phoneme sequences pronounceable by monolingual voices. The method is efficient, language independent, entirely phonetics-based and it enables any Loquendo TTS voice to speak all the languages provided by the system.

Traditional systems are conceived to read monolingual texts; multilingual texts can be correctly read by changing the voice at every language change. This can be unfeasible for truly mixed-language texts, where changes occur frequently and are embedded in sentences and phrases. Real applications require more flexibility to handle a variety of situations: texts coming from different sources in unpredictable language (e.g. internet), e-mails or office documents written in more than one language, foreign names or phrases (e.g. film titles) within information services.

The optimal solution would be to have the same TTS voice reading the whole mixed-language text, applying an automatic phonetic transcriber for the foreign language and then mapping the obtained transcription onto the phonemes of the native language of the voice, in order to access its acoustic units.

This approach brings an "approximate pronunciation". Looking at many real cases, although this is an approximate approach, may fit better to reality. In fact, a speaker having to pronounce foreign words included in a text written predominantly in his or her own language will be generally inclined to pronounce these words in a manner that may differ - also significantly - from the correct pronunciation of the same words when included in a complete text in the corresponding foreign language. The approximation of this kind of pronunciation is especially due to the speaker choice of maintaining his native-tongue phonological system. This choice is due to co-articulation, economy of effort and also to psychosocial factors, as adopting the correct pronunciation may be regarded as an undue sophistication and, as such, rejected in common usage.

Loquendo Language Guesser makes it possible to identify the different languages contained within any kind of document. Identifying a language by means of a text is an extremely complex task to achieve. Complexity increases significantly as the number of recognizeable languages grows. And the briefer the text, the greater the likelihood of increased ambiguity there is.

Loquendo's Language Guesser module used in conjunction with Loquendo TTS synthetic speech, currently enables the identification of the following languages: English, Spanish, French, Brazilian Portuguese, German, Italian, Swedish, Catalan, Greek and Dutch. With Loquendo Language Guesser, systems integrators can now create applications that are capable of reading a document containing text in a variety of languages - always in the appropriate language.

LoquendoTTS can guess the language of a chunk of text, but in order to get the automatic language detection, you need to have installed the CD “Mixed Language Capabilities” (optional).

The automatic guessing can be enabled using the control tags, or with an appropriate API call (see LoquendoTTS Programmer’s Guide for details), no matter of the API set used (tts or SAPI). Two different modes are possible:

1. Language Switch
2. Voice Switch

In mode 1) the language is automatically changed, without switching the active voice. For instance, the American English voice "Dave" can switch temporarily to French, and use the French rule set, in order to pronounce a French sentence, and then come back to English. The French pronunciation is less accurate than a French voice's one: it sounds more like an English native speaker that speaks French.

In mode 2) the voice is changed automatically, choosing the most appropriate one among the installed voices. In case more than a voice is present, speaking the same language, here is the precedence:

1. Among the open voices (already loaded in memory), finds for a voice of the desiderated language, with the same sex of the currently active voice
2. Among the open voices (already loaded in memory), finds a voice of the desiderated language
3. Finds an installed voice (not already loaded in memory) of the desiderated language, with the same sex of the currently active voice
4. Finds an installed voice (not already loaded in memory) of the desiderated language

If Loquendo TTS cannot find a voice to perform the voice switching, the command is ignored.

The automatic guessing uses the Language Guesser to detect the language; the application must define the length of the part of speech the guessing must be applied to, among:

1. Paragraph by Paragraph
2. Sentence by Sentence
3. Phrase by Phrase
4. Word by Word

"Phrase by Phrase" and "Word by word" modes make sense only combined with the Language Switch, whilst the other two modes can be applied both to Language and Voice Switches.

Finally, in order to facilitate the Language Guesser job, it is possible to define the list of languages to guess among.

In order to activate and configure the Language Guesser, a specific control tag can be added to the text: "\@AutoGuess=<type>:<language list>". For a more detailed information about this configuration command, see the ""\@AutoGuess=<type>:<language list>" description in the "Control tags" section.

Note that "Word by word" mode may sometimes lead to unpredictable results, due to intrinsic ambiguity of most words. For instance the sentence "Mission impossible" can be either English or French. The guessing would be more accurate when applied to a longer part of speech.

In order to avoid this kind of unpredictable results, it is always possible to force the language switch directly inside the text, using the "\lang=<mnemonic>" tag, where the "<mnemonic>" string is the name of a language. For a more detailed information about the language switch command, see the "\lang=<mnemonic>" description in the "Control tags" section.

Here you can find the list of language mnemonics (LoquendoTTS proprietary), followed by language mnemonic (similar to standard used by SSML), sublanguage menmonics (similar to standard used by SSML) and eventual one or more other LoquendoTTS proprietary mnemmonics:

```
Catalan: ca,ca-ES,Catalan
Chinese: zh,zh-CN,CN,Mandarin,Chinese
Dutch: nl,nl-NL,Dutch
English: en,en-GB,GB,British,EnglishGb
English: en,en-US,US,American,EnglishUs
French: fr,fr-FR,French
German: de,de-DE,German
Greek: el,el-GR,Greek
```

```
Italian: it,it-IT,Italian
Portuguese: pt,pt-BR,BR,Brazilian,PortugueseBr
Portuguese: pt,pt-PT,PortuguesePt
Spanish: es,es-AR,ar,SpanishAr,Argentine
Spanish: es,es-CL,CL,Chilean,SpanishCl
Spanish: es,es-ES,SpanishEs,Castilian
Spanish: es,es-MX,mx,SpanishMx,Mexican
Swedish: sv,sv-SE,Swedish
Italian: it,it-IT,Italian
```

Lowercase version of the first column mnemonics can be used too.
When more than a sublanguage is available, as in English where we have EnglishGB and EnglishUS, if a “\lang=English” control tag is activated to enable English phonetic mapping on a previous different language, the “EnglishGB” sublanguage is selected by default. The default for Spanish is the “Mexican” sublanguage, and the default for Portuguese is the “Brazilian” sublanguage.
In order to change the selection from these default, another sublanguage can be activated; for example: “\lang=EnglishUs”.

5 Control tags

N.B. The following information applies to the legacy interface. If the Speech API 4.0 or 5.1 interfaces are used, the commands must be given as described in the Microsoft SAPI documentation.

Commands modifying the Loquendo™ TTS playback parameters can be inserted in the text. Such commands are preceded by a backslash **'\'** and act on the following word or until a command is given which cancels their effect. Command specifications may be changed in future versions of Loquendo™ TTS. More than one command can be given in a single control tag as in:

\tag1<parameters>\tag2<parameters>

A tag sequence must ALWAYS be followed by a space (SPACE, TAB, RETURN, NEWLINE, FORMFEED) AND THEN followed by a word. The only exception is the command \f "phonetic transcription" which does not require any additional word.

The commands described below, and those for speaking rate and tone in particular, should be used with great care. The default values will usually provide the best results.

5.1 Voice change

\voice=<mnemonic> (or the obsolete: \!<mnemonic>)*	Voice change. This tag forces a voice switch among the voices. The mnemonic must be the name of an installed voice. This is a way to allow voice changing by means of a synchronous text-embedded command. **Pay attention: this tag set to their default values the prosodic parameters: speaking rate, tone and volume.** (see also *ttsNewVoice* API in the Loquendo™ TTS Programmer's guide for details).

Example:

\voice=Paola ciao. **\voice=**Susan hello. ("ciao" is read by the voice "Paola", then "hello" is read by the voice "Susan").

5.2 Language change

\lang=<mnemonic>	Set foreign language. This tag forces a language switch among the opened languages. The mnemonic must be the name of a previously opened language. This is a way to allow language changing without changing the voice. So the Speaker is able to speak foreign . If the "Mixed Language Support" has been installed, the switch can happen between all the LoquendoTTS languages (not only the opened ones). Valid "<mnmenonic>" can be: "english", "french", "german", "italian", "spanish", "greek", "swedish", "portuguese", "catalan", "chinese" and "dutch", but other standard mnemonics are allowed. For more information about this tag and for other valid language mnemonics, see the "Mixed Language Support (optional)" chapter.
\lang=	Reset native language. This is a the language change reset: go back to the initial language.

Examples:

In Italian "true or false" is **\lang=**italian "vero o falso" **\lang=** .
(English example where the pronounce of "vero o falso" is improved activating the italian phonetic mapping. The last control tag reset the language to English phonetics again)

In Inglese "vero o falso" si dice **\lang=**english "true or false" **\lang=** .
(Italian example where the pronounce of "true or false" is improved activating the english phonetic mapping. The last control tag reset the language to Italian phonetics again)

5.3 Language guesser configuration

\@AutoGuess=<type>:<language list>	Language guesser configuration. This tag activate and configure the Language Guesser. It can be used only if the "Mixed Language Support" has been installed (it is a separate optional CD-ROM). For more information about the Language Guesser, see the "Mixed Language Support" chapter.The "<type>" string must be one of the following: **"no" –** no AutoGuess mode **"VoiceParagraph" –** Detects language and changes voice accordingly *paragraph by paragraph* **"VoiceSentence" -** Detects language and changes voice accordingly *sentence by sentence* **"VoicePhrase" -** Detects language and changes voice accordingly *phrase by phrase* **"LanguageParagraph" –** Detects and change language *paragraph by paragraph* without changing the active voice **"LanguageSentence" –** Detects and change language *sentence by sentence* without changing the active voice **"LanguagePhrase" –** Detects and change language *phrase by phrase* without changing the active voice **"LanguageWord" –** Detects and change language *word by word* without changing the active voice **"BothParagraphSentence" –** Combines the effects of "**VoiceParagraph"** and "**LanguageSentence"** **"BothParagraphPhrase" –** Combines the effects of "**VoiceParagraph"** and "**LanguagePhrase"** **"BothParagraphWord" –** Combines the effects of "**VoiceParagraph"** and "**LanguageWord"** **"BothSentencePhrase" –** Combines the effects of "**VoiceSentence"** and "**LanguagePhrase"** **"BothSentenceWord" –** Combines the effects of "**VoiceSentence"** and "**LanguageWord"** **"BothPhraseWord" –** Combines the effects of "**VoicePhrase"** and "**LanguageWord"**

The "<language list>" can be one or more language names separated by commas, where the languages can be: "english", "french", "german", "italian", "spanish", "greek", "swedish", "portuguese", "catalan" and "dutch", but other standard mnemonics are allowed.
For more information about this tag and for other valid language mnemonics, see the "Mixed Language Support (optional)" chapter.
For the last six types (the "Both…" ones) a postponed '-' (minus) character after the language name (e.g. "swedish-") means that voice changes are admitted, but not "language only" changes.
A prefixed '-' (minus) means that only language changes are admitted (not voice changes).

Some basic examples:

`\@AutoGuess=VoiceSentence:Italian,English` (sentence by sentence changes among Italian and English voices)

`\@AutoGuess=BothSentenceWord:French-,Spanish-,English` (sentence by sentence detects the right language and changes voice accordingly. In addition, while speaking with non-English voices, English words are detected and pronounced with the English phonetic rule set).

Another example:

\voice=Susan hello.
\@AutoGuess=no:italian,english
A true English sentence .
Una vera frase Italiana .

(The Language Guesser is not active, so every sentence will be read by the voice Susan with English pronounce)

\@AutoGuess=LanguageSentence:italian,english
A true English sentence .
Una vera frase Italiana .

(The Language Guesser is active, so every sentence will be read by the voice Susan, but with Italian phonetic mapping for the second sentence)

\@AutoGuess=VoiceSentence:italian,english
A true English sentence .
Una vera frase Italiana .

(The Language Guesser is active, and the voice switch too, so the first sentence will be read by the voice Susan, but the second with an Italian voice and Italian pronounce)

5.4 User lexicons

\lexicon=<filename>	User lexicon load. This tag allows to load a new lexicon for the current voice; it is possible to load many lexicons. The last loaded lexicon will be accessed first, overriding the others in case of conflicting definitions. The filename can contain only slashs in order to specify a full path (backslashes are not admitted, thus the syntax will be UNIX like, even if you are in the Windows environment). Also the blanks are not admitted inside the path, so a string "%20" must be used in place of each blank. The <filename> can be an URL too (supported on Windows, on Linux by means of the library "libcurl.so" usually included in the Linux distributions, not supported on Solaris).
\lexicon=*<filename>	User lexicon unload. Unload the lexicon named <filename>, so to unload a lexicon file use the star character "*" before the filename (after equal symbol).
\lexicon=	Unload the last user lexicon (no filename need to be specified).

Examples:

If a personal lexicon named "new.lex" is created, containing this example expansion:
"hw" = "hardware", the lexicon can be loaded with the following:

\lexicon=c:/temp/new.lex

and the sequence "hw" will be read as "hardware".
In order to go back to the previous situation, the lexicon can be unloaded with the following:

\lexicon=*c:/temp/new.lex

If another personal lexicon is named "another new.lex", with a blank inside the name, it can be loaded with the following:

\lexicon=c:/temp/another%20new.lex

5.5 Plugin lexicons

\plugin=<mnemonic>	Plugin lexicon load. This tag allows to load a specialized plugin lexicon for the current voice. It is possible to load many plugin and user lexicons. The last loaded lexicon will be accessed first, overriding the others in case of conflicting definitions. For the list of the mnemonics of the available lexicons, for a given language, see the relative Language Reference Guide (inside the voice CD-ROM distribution) or the TTSDirector "Effects" menu.
\plugin=*<mnemonic>	Plugin lexicon unload. Unload the plagin lexicon named <mnemonic>.

Examples:

If a plugin SMS lexicon is available for the active language (containing expansions for SMS typical abbreviations), the lexicon can be loaded with the following:

\plugin=SMS

In order to go back to the original situation, the lexicon can be unloaded with the following:

\plugin=*SMS

5.6 Numbers say as

\Nr	Say as cardinal the next digit string. In other words, marks the following word or token as a cardinal number (amount or currency). This can be used to change default Loquendo™ TTS behavior in the following cases: • big sequence of digits (that are normally interpreted as telephone numbers) • roman numbers (that are normally read as letters)
\Nm or \Nf (feminine)	Say as (masculine or feminine) ordinal the next digit string. In other words, marks the following word or token as an ordinal number. This can be used to change default Loquendo™ TTS behavior in the following cases: • big sequence of digits (that are normally interpreted as telephone numbers) • roman numbers (that are normally read as letters) Two different tags are provided because in some languages (for instance Spanish or Italian) ordinal numbers can be masculine or feminine. The following control tags have the same effect, but permanent (on all next digit strings): **\@DefaultNumberType=MasculineOrdinal** **\@DefaultNumberType=FemininеOrdinal**
\Nt	Say as telephone number the next digit string. In other words, marks the following token as a telephone number. This can be used to change default Loquendo™ TTS behavior reading of comma-delimited sequences of digits (that are normally interpreted as amounts). The way in which telephone numbers are read depends on the language. The following control tag has the same effect, but permanet (on all next digit strings): **\@DefaultNumberType=telephone**
\Nx	Say as a code number the next digit string. In other words, marks the following token as a code number. This can be used to change default Loquendo™ TTS behavior reading of comma-delimited sequences of digits (that are normally interpreted as amounts). Code numbers are read digit by digit. The following control tag has the same effect, but permanent (on all next digit strings): **\@DefaultNumberType=code**
\Nh	Say as a time the next digi string. In other words, marks the following token as a time. The following control tag has the same effect, but permanent (on all next digit strings): **\@DefaultNumberType=hour**
\@DefaultNumberType=generic	Reset all permanent modifiers (like \@DefaultNumberType=MasculineOrdinal, \@DefaultNumberType=telephone, …).
\Nd<format>	Date format. The date will be interpreted and pronounced according to a format, where the **<format>** can be: “mdy” (month day year), “ymd”, “ym”, “my”, “md”, “y”, “m”, “d” ,as for SSML say-as date tag.
\Nd	Reset date format. (Reset the \Nd<format> tag).

Examples:

253126 . **\Nr** 253126 .
(In English, the first number is intepreted by TTS as a phone number, so is read digit by digit. The same number after \Nr is forced to be read as a cardinal number).

1. **\Nm** 1.
(In englishUS, the first number is read "one", the second is read as "first", that is its ordinal version)

1 . **\Nm** 1 . 2.
1 . **\@DefaultNumberType=MasculineOrdinal** 1 . 2.
(In englishUS, the first number is read "one", the second is read as "first", and the third as "second", but only in the second example, because only the \@DefaultNumberType=MasculineOrdinal has a permanent effect)

1. **\Nf** 1.
(In Italian is read as "uno prima", because "prima" is the feminine ordinal version of the number "1").

25000 . **\Nt** 25000 .
(In English, the first number is read as a cardinal number. The same number after \Nt is forced to be read digit by digit as a phone number).

67890. **\Nx** 67890.
(The first number is read as a big integer, the second digit by digit)

1990 . **\Ndy** 1990 . \Nd 1990.
10-1990. **\Ndmy** 10-1990. **\Nd** 10-1990.
(In these two examples, the first number sequence is not recognized and pronounced as a date; the second is pronounced as a date because it is forced by the control tag; the third sequence is read as the first one, because the "\Nd" tag reset the previous one)

5.7 Phonetic input

\f<phonemes>	Insert phonemes. This tag allows to give the phonetic transcription of a word instead than its graphemic form. Phonemes must be separated by an hyphen (a "-" character). See Working with Lexicon chapter too for more informations.
\ipa=<ipastring>	Insert IPA phonemes. This tag allows to give the IPA (International Phonetic Alphabet) string phonetic transcription of a word instead of its graphemic form. Use a "%20" as separator between the phonetic transcription of different words.
\SAMPA= <proprietary>; <phonemes>	Insert SAMPA phonemes. This tag allows to give the SAMPA string phonetic transcription of a word instead than its graphemic form. **<proprietary>** is a string that defines a specific version (proprietary) of SAMPA. This string is optional; the only values allowed are "NAVTEQ" and "TELEATLAS". NAVTEQ™ and TELEATLAS™ are registered trade marks. If the <proprietary> string is omitted, the standard UCL SAMPA conventions will be used, according to the phoneme tables from: http://www.phon.ucl.ac.uk/home/sampa/ **<phonemes>** is a string of SAMPA phonemes, with no blank inside, used as the phonetic input of the TTS. This string is mandatory, and this kind of phonetic input is provided only for isolated words or short utterances (like placenames). Please **use a '#' character instead of the blank** character, if the original SAMPA string has one or more blanks inside. A syllabic separator is **mandatory** for all the polysyllabic transcriptions. This character could be different for specific <proprietary> versions. **Also for the UCL SAMPA**, **a mandatory** syllabic separator '|' must be used, which is not part of the original UCL SAMPA standard. **Warning**: only SAMPA phonemes belonging to "Italian", "French", "Castilian", "German", "EnglishGb", "EnglishUs", "Dutch" and "PortuguesePt" languages are currently supported. **Warning**: secondary stress, which in SAMPA is the '%' character, is presently converted into a primary stress ('"' in SAMPA). In order to simply skip the secondary stress, set to NO the registry key "SampaSecondAccent" (for more information, see the LoquendoTTS Programmer's Guide.)

See the specific Language Reference Guides for the list of valid phonemes in the different formats. For additional information, see the "Working with Lexicon" chapter.

Please note that this TTS software allows you to use both Loquendo TTS phonemes symbols, SAMPA phonemes symbols as well as IPA symbols, but the first two are simpler to enter, because they have been designed using only ASCII characters.

Instead, when entering IPA symbols, you have to enter them in UNICODE and more specifically you have to use one of the following syntaxes (borrowed from the HTML world):

- "&#D;" where D is a decimal number;
- "&#xH;" or "&#XH;" where H is a hexadecimal number.

At the following link http://www.phon.ucl.ac.uk/home/wells/ipa-unicode.htm you can find the correspondence map between IPA-UNICODE.
You can also look at http://www.unicode.org/charts/PDF/U0000.pdf and http://www.unicode.org/charts/PDF/U0250.pdf.

For more information about SAMPA phonemes, you can refer to the traditional WEB site of the UCL – University College London: http://www.phon.ucl.ac.uk/home/sampa/, where a general description and detailed phonetic tables are included.

EnglishUS language example:

hello . **\f**h-HEh-l-`HOU . **\ipa=**həlˈoʊ .
(the same EnglishUS word, in three input versions: ortographic, phonetic with LoquendoTTS symbols, phonetic with IPA symbols)

Italian language examples:

ciao. **\f**T$-`a-o . **\ipa=**ʧˈao .
(the same Italian word, in three input versions: ortographic, phonetic with LoquendoTTS symbols, phonetic with IPA symbols)

\fm-`a-m:-a .
\ipa=mˈamːa .
\ipa=mˈamːa .
(the Italian word "mamma" in three different, but equivalent, phonetic transcriptions).

Some Italian language SAMPA examples:

\SAMPA=to|"ri|no .
("Torino" in SAMPA phonemes)

\SAMPA="san#dZo|"van|ni .
("San Giovanni" in SAMPA phonemes)

Some French language SAMPA examples:

\SAMPA=aR|"si .
("Arcy" in SAMPA phonemes)

\SAMPA=%le#"gRa~Z .
("Les Granges" in SAMPA phonemes)

\SAMPA=NAVTEQ;i|vER|"ni .
("Iverny" in SAMPA phonemes according to a proprietary "NAVTEQ" version; NAVTEQ™ is a registered trade mark.)

\SAMPA= TELEATLAS;IvER"ni .
("Iverny" in SAMPA phonemes according to a proprietary "TELEATLAS" version; TELEATLAS™ is a registered trade mark.)

5.8 Spelling

\s	Spell out next word. The following word is pronounced letter by letter[3].
\s0	Never spell out. Every following word, including acronyms, is pronounced as a non-spelled word. The following control tag has the same effect: **\@SpellingLevel=pronounce**
\s1	Standard reading mode. The following control tag has the same effect: **\@SpellingLevel=normal**
\s2	Spell out every word. (Every following word is spelled out). The following control tag has the same effect: **\@SpellingLevel=spelling**

Examples:

Please give us your us phone number.
(wrong, because the second “us” is pronunced as the first)
Please give us your **\s** us phone number.
(right, because the second “us” is spelled letter by letter)
Please give us your **\s2** us phone number.
(wrong, because not only the second “us” is spelled letter by letter, but “phone number” too)
Please give us your **\s2** us **\s1** phone number.
(right, because only the second “us” is spelled letter by letter, but for a single word a “\s” it is enough)
Please give US your **\s** us phone number.
(wrong, because the first “US” is interpreted as United States and spelled out)
Please give **\s0** US your **\s** us phone number.
(right, because the first “US” is not spelled out, and the second is spelled)

5.9 Read (aloud) punctuation

\sp1	Read (aloud) punctuation. The punctuations following this tag are read (aloud) up to a “\sp0” tag.
\sp0	Do not read (aloud) punctuation. The punctuations following this tag are not read (aloud).

Examples:

This is a **\sp1** . inside a sentence **\sp0** .
(the TTS says: “this is a dot inside a sentence”: the first dot is read aloud, while the second not, because is intepreted as standard punctuation)

[3] Spelling out is necessary for playing back certain acronyms correctly. At the moment, the system automatically spells out only those acronyms that consist entirely of consonants. For example, L’azienda svedese RIV SKF … is pronounced correctly as *“l’azienda svedese riv esse cappa effe”* while the system would render Il colosso informatico IBM as *“Il colosso informatico ibm”* , where IBM is pronounced as if it were a word. To produce a correct pronunciation, we must thus insert the command \s in the sentence: Il colosso informatico \s IBM. This yields the correct result *“Il colosso informatico ì bì èmme”*.

5.10 Read (aloud) control tags

\@TaggedText =false	Read (aloud) control tags. All control tags are not processed but pronounced up to the next "\{@TaggedText=true" tag.
\{@TaggedText =true	Do not read (aloud) control tags. All control tags are processed and not pronounced (this is the default mode).

Example:

This is the **\Nm** 1 . \@**TaggedText=false** This is the **\Nm** 1 . **\{@TaggedText=true** This is the **\Nm** 1 . (This sentence is pronounced "This is the first. This is the backslash n m 1. This is the first.", because every tag between "\@TaggedText=false" e "\{@TaggedText=true" is read aloud)

Warning: Please note the special characters sequence "\{@", used when setting TaggedText to true. This is a special sequence designed to re-enable properly the control tag processing features.

5.11 Prosodic pauses

\Pp	Enable breath pause insertion. (That is some prosodic pauses are inserted inside sentences). This is the default behavior.
\Pm	Breath pauses only at punctuation. Disables the prosodic pauses insertion (no prosodic pauses are inserted inside text: only punctuation marks produce pauses)
\Pw	Read word by word. (Enables words by words reading and it is disabled by the tag "\Pp").
\@MultiCRPause =false	Do not insert breath pauses at empty lines. (Usually empty lines in text generate a pause. If you set this parameter to "false", no pause is generated).
\@MultiCRPause =true	Insert breath pauses at empty lines. (Usually empty lines in text generate a pause. If you set this parameter to "true", pause is generated – this is the default).
\@MultiSpacePause =false	Do not insert breath pauses at multiple spaces or tabs. (Usually multiple spaces or tabs in text generate a pause. If you set this parameter to "false", no pause is generated)
\@MultiSpacePause =true	Insert breath pauses at multiple spaces or tabs. **(**Usually multiple spaces or tabs in text generate a pause. If you set this parameter to "true", pause is generated – this is the default).
\@MaxParPause =<value>	Insert breath pauses at titles. Usually lines short than 5 words (like titles or signatures) are automatically terminated by a pause. You can change <value> from 5 to a different value; use "0" (zero) if you want to disable this feature.

Examples:

In questa lunga frase viene inserita una pausa.
\Pm In questa lunga frase viene inserita una pausa.
\Pp In questa lunga frase viene inserita una pausa.
(In the first Italian language example, a breath pause is automatically inserted just before the word "viene", in order to improve the prosody of the sentence. This automatic insertion is disabled by the "\Pm" tag in the second example, so no pause is done, while the pause is pronounced again in the third example, because the "\Pp" tag restore the default condition).
(The automatic breath pause insertion is available only for some languages, like Italian).

\Pw Now pausing at every word. **\Pp** Standard reading again.
(The first sentence is read word by word, while the second is read in the standard way, with no pause between the words, as in the following "Now. Pausing. At. Every. Word. Standard reading again.").

\@MultiCRPause=false
Thank you

Best regards
(In this example, no pause is inserted between "Thank you" and "Best regards", so it sounds quite innatural).

\@MultiCRPause=true
Thank you

Best regards
(In this example, a pause is inserted between "Thank you" and "Best regards", so it sounds more natural than the previous example – This is the default behaviour).

\@MultiSpacePause=false
Thank you Best regards
(In this example, no pause is inserted between "Thank you" and "Best regards", so it sounds quite innatural).

\@MultiSpacePause=true
Thank you Best regards
(In this example, a pause is inserted between "Thank you" and "Best regards", so it sounds more natural than the previous example – This is the default behaviour).

\@MaxParPause=4
The Whole Story
Chapter one
(In this example, a pause is inserted between "The Whole Story" and "Chapter one", because with the "4" value the line shorter than 4 words are interpreted as a separate title).

\@MaxParPause=0
The Whole Story
Chapter one
(In this example, no pause is inserted between "The Whole Story" and "Chapter one", because with the "0" value no line is interpreted as a separate title).

5.12 Prominence

\u<word>	Unstress a word. (The following <word> will have no stress, like many functional words inside a sentence).

5.13 Emphasis

\emphasis+	Increase. This tag increases the speech emphasis with a triple volume increase (treble \volume+), a triple pitch increase (treble \pitch+) and a double speed decrease (twice \speed-).
\emphasis-	Decrease. This tag reduces the speech emphasis with a triple volume decrease (treble \volume-), a treble pitch decrease (treble \pitch-) and a double speed increase (twice \speed+).
\emphasis	Reset. This tag resets emphasis to the default values.

5.14 Punctuation pause

\p<insert milliseconds> <insert punctuation>	Duration (in msec). Assigns duration in milliseconds to the punctuation symbol which follows. Punctuation can be “. ; : ! ? , ”.

Examples:

This is a long **\p**3000 ! pause inside a sentence.

(A 3 seconds pause is inserted after the “long” word).

5.15 Speaking rate

\speed=<num> (or the obsolete: \v<num>)*	Percentage change. This tag changes speaking rate from the following word to the next command; <num> is expressed in percentage and ranges from a minimum of 0 to a maximum of 100. The range of the speaking rate can be modified by using \SpeedRange (or the obsolete: \VR) tag. Pay attention: up to the previous 6.3.x versions, the range was 0 to 10; it is possible to restore this behaviour by setting this key: OldProsodyRange=yes – for more information, see the LoquendoTTS Programmer's Guide.
\speed+ (or the obsolete: \v+)	Increase. This tag increases the current speaking rate by 10 words per minute.
\speed- (or the obsolete: \v-)	Decrease. This tag reduces the current speaking rate by 10 words per minute.
\speed (or the obsolete: \v)	Reset. This tag resets speaking rate to the default value.

Examples:

\speed=<num>

This text should be spoken at the default speed.
\speed=0 This text should be spoken at the minimum speed.
\speed=50 This text should be spoken at the default speed.
\speed=100 This text should be spoken at the maximum speed.
\speed This text should be spoken at the default speed.
(The text of this example is self-explanatory)

\speed Normal speed . **\speed+** A bit faster **.** **\speed+** Faster . **\speed+** **\speed+** **\speed+** Very fast .
\speed Normal speed . **\speed-** A bit slower . **\speed-** Slower . **\speed-** **\speed-** **\speed-** Very slow .
(The text of this example is self-explanatory; the increase or decrease steps are of limited range)

*Obsolete control tags will be removed in the next releases.

5.16 Tone (fundamental frequency)

\pitch=<num> (or the obsolete: \t<num>)*	Percentage change. This tag changes tone from the following word to the next command; <num> ranges from a minimum of 0 to a maximum of 100. The range of the pitch is dimensionless and can be modified by using \PitchRange (or the obsolete \TR) tag. Pay attention: up to the previous 6.3.x versions, the range was 0 to 10; it is possible to restore this behaviour by setting this key: OldProsodyRange=yes – for more information, see the LoquendoTTS Programmer's Guide.
\pitch+ (or the obsolete: \t+)	Increase. This tag increases the current tone by 1 semi-tone.
\pitch- (or the obsolete: \t-)	Decrease. This tag reduces the current tone by 1 semi-tone.
\pitch (or the obsolete: \t)	Reset. This tag resets tone to the default value.
\m<num>	Monotonous. This tag set pitch to <num> in Hz, giving the effect of a monotonous voice. It works only with Italian Mario and Sonia voices.

Examples:

This text should be spoken at the default pitch.
\pitch=0 This text should be spoken at the minimum ptich.
\pitch=50 This text should be spoken at the default pitch.
\pitch=100 This text should be spoken at the maximum pitch
\pitch This text should be spoken at the default pitch.
(The text of this example is self-explanatory)

\pitch Normal pitch . **\pitch+** A bit higher . **\pitch+** Higher . **\pitch+ \pitch+ \pitch+** Very high .
\pitch Normal pitch . **\pitch-** A bit lower . **\pitch-** Lower . **\pitch- \pitch- \pitch-** Very low .
(The text of this example is self-explanatory; the increase or decrease steps are of limited range)

*Obsolete control tags will be removed in the next releases.

5.17 Volume (gain)

\volume=<num> (or the obsolete: \V<num>)*	Percentage change. This tag changes volume from the following word to the next command; <num> is expressed in percentage and ranges from a minimum of 0 to a maximum of 100 (200 with the obsolete \V<num>). The range of the volume is dimensionless and can be modified by using \VolumeRange tag. Pay attention: up to the previous 6.3.x versions, the range was 0 to 10; it is possible to restore this behaviour by setting this key: OldProsodyRange=yes – for more information, see the LoquendoTTS Programmer's Guide.
\volume (or the obsolete: \V)	Reset. This tag reset the volume to the default value.

Examples:

This text should be spoken at the default volume.
\volume=0 This text should be spoken at the minimum volume.
\volume=50 This text should be spoken at the default volume.
\volume=100 This text should be spoken at the maximum volume.
\volume This text should be spoken at the default volume.
(The text of this example is self-explanatory – pay attention: with "\volume=0", nothing can be heard)

*Obsolete control tags will be removed in the next releases.

5.18 Prosody change range

\SpeedRange=<min,med,max> (or the obsolete: \VR<min,med,max>)*	For speed. This tag changes speed range, defining minimum, maximum and central values; this command affects the speaking rate tag behavior. This command is useful to map physical prosody values (words per minute) to a predefined scale (for instance in designing slide controls for GUI applications). For instance, the command \SpeedRange=0,5,10 defines a speed range from 0 to 10, with 5 as central value. After this command the tag "\speed=10" will lead speed to its maximum, while "\speed=0" will lead it to its minimum. You can change from a dimensionless range to a physical one by the command \SpeedRange=0,0,0 followed by a new range definition. In this case minimum, maximum and central values will be expressed as words per minute.
\PitchRange=<min,med,max> (or the obsolete: \TR<min,med,max>)*	For pitch. This tag changes pitch range, defining minimum, maximum and central values; this command affects the tone tag behavior. This command is useful to map physical prosody values (hertz) to a predefined scale (for instance in designing slide controls for GUI applications). For instance, the command \PitchRange=0,5,10 defines a pitch range from 0 to 10, with 5 as central value. After this command the tag "\pitch=10" will lead pitch to its maximum, while "\pitch=0" will lead it to its minimum. You can change from a dimensionless range to a physical one by the command \PitchRange=0,0,0 followed by a new range definition. In this case minimum, maximum and central values will be expressed as hertz.
\VolumeRange=<min,med,max>	For volume. This tag changes volume range, defining minimum, maximum and central dimensionless values; this command affects the volume tag behavior. This command is useful to map physical prosody values to a predefined scale (for instance in designing slide controls for GUI applications). For example, the command \VolumeRange=0,50,100 defines a volume range from 0 to 100, with 50 as central value. After this command the tag "\volume=100" will lead volume to its maximum, while "\volume=0" will lead it to its minimum.

Examples:

This text should be spoken at the default speed.
\speed=0 This text should be spoken at the minimum speed.
\speed=50 This text should be spoken at the default speed.
\speed=100 This text should be spoken at the maximum speed.
\speed This text should be spoken at the default speed.
(Set of examples according to the default speed range)

\SpeedRange=0,5,10
This text should be spoken at the default speed.
\speed=0 This text should be spoken at the minimum speed.
\speed=5 This text should be spoken at the default speed.
\speed=10 This text should be spoken at the maximum speed.
\speed This text should be spoken at the default speed.
(Set of examples according to the new default speed range - the results on the voice are the same)

More details:

Loquendo TTS cannot currently change the "pitch shape" of a voice, but it may only "shift the pitch" up and down of a certain small quantity that is different from a speaker to another (without introducing too much distortion).

As consequence of that, it is not possible to have monotonic voices (you could think to write \PitchRange=0,0,0 - this is WRONG!).

Normally when you use the \pitch tag, you can make a voice speaking with a tone more or less high.

As usually the pitch values are bound to a sliding cursor (in graphical interfaces, such us our Edit2Speech and TTSDirector), Loquendo has introduced the control tag \PitchRange to specify the figures you may use as minimum, average (default), maximum. So, if an interface uses the values 0, 5, 10, you may impose the same values on Loquendo TTS (that by default uses 0, 50, 100). When you set \pitch=0 you set the minimum pitch that such voice can use and when set \pitch=10 you set the maximum pitch. \pitch=5 or \pitch (alone) set the default pitch. Values beyond such values are clipped to the range imposed.

We decided to use "pure" figures (without any measure, i.e. "dimensionless" figures) because if we'd used for example Hertz, by changing from a voice to another you'd get unpredictable results.

By using "pure" figures, the minimum is always the same regarding the voice (and the same for maximum and average/default).

Please note that the Edit2Speech and TTSDirector interfaces use the ranges 0, 50, 100 so, if you change the ranges, the slider is no more synchronised with the actual pitch (because it may be out of scale).

If you set \PitchRange=0,0,0 you renounce to set the pitch with "pure figures" and you move to the Hertz field. This is deprecated, because the baseline Hertz values are different for each voice. E.g. Elizabeth has the following baseline values: "110,150,250".

If with \PitchRange=0,0,0 you try to use \pitch=50, actually you set it to 110, that is the minimum allowed for Elizabeth (you cannot go beyond the minimum and the maximum values).

We suggest to never use the \PitchRange=0,0,0 feature unless you have a "scientific" purpose to achieve.

Examples:

\voice=Elizabeth The following test will be read by Elizabeth

\PitchRange=0,5,10
\pitch This text should be spoken at the default pitch.
\pitch=0 This text should be spoken at the minimum pitch.
\pitch=5 This text should be spoken at the default pitch.
\pitch=10 This text should be spoken at the maximum pitch.
\pitch This text should be spoken at the default pitch.

\PitchRange=0,0,0

\pitch This text should be spoken at the default pitch (150 Hz).
\pitch=150 This text should be spoken at the default pitch (150 Hz).

\pitch=0 This text should be spoken at minimum pitch (110 Hz).

\pitch=80 This text should be spoken at minimum pitch (110 Hz).

\pitch=130 This text should be spoken at pitch 130 Hz
\pitch=200 This text should be spoken at pitch 200 Hz

\pitch=250 This text should be spoken at maximum pitch (250 Hz).
\pitch=500 This text should be spoken at maximum pitch (250 Hz).

*Obsolete control tags will be removed in the next releases.

5.19 Duration control

\dur=<msec>	Force duration. This tag forces the synthesis duration (expressed by <msec> in milliseconds) for the following text, until a mandatory "\durEnd" tag. Important note: the text included between "\dur=…" and "\durEnd" tags must not include pauses and punctuation marks; it is recommended to use "\Pm" tag before this tag to disable prosodic pauses. The <msec> value must be at least the 30% of the speaking time between "\dur" and "\durEnd" tags, otherwise there will be no effect.
\durEnd	End force duration. This tag must be used to define the end of text with duration control.

Examples:

This is standard reading .
\dur=600 This is a fast reading **\durEnd** .
\dur=2000 This is a slow reading **\durEnd** .
(In the second example, the duration of the sentence is imposed to 600 msec, resulting in a very fast reading. In the third example, the duration of the sentence is imposed to 2000 msec, resulting in a very slow reading.)

5.20 Raw signal files playing

\w<filename>	Play. This tag allows playing of a RAW signal file at the specified position in the text. The filename can contain only slashs in order to specify a full path (backslashes are not admitted, thus the syntax will be UNIX like, even if you are in the Windows environment). Also the blanks are not admitted inside the path, so a string "%20" must be used in place of each blank. The signal file must have no header and use the same coding and the same sampling frequency as the TTS; the file must have a Little Endian (Intel) byte order.

Examples:

To play a file named "new.raw":

\wc:/temp/new.raw

To play a file named "another new.raw", with a blank inside the name:

\wc:/temp/another%20new.raw

5.21 Audio mixer capabilities

\audio(command [;command;…])	This tag allows sending commands to the Audio Mixer. Writing more commands separated by a ';' is allowed.

The *audio mixer* allows mixing sound files and voice. It's possible to mix one or more sound files **simultaneously,** at the same time. Every sound file (audio source) is considered as an independent audio track, with independent volume, timeline and sample rate.
The sample rate frequency of the audio sources is automatically converted according to the voice frequency used. The *audio mixer* supports 16 bit sound files, mono and stereo, with arbitrary sample rate frequency.
". wav" files are supported and played.
".mp3", ".wma", ".asf", ".ogg", ".avi", ".mpg" are not supported and are not played.
". raw" , ".pcm" and any other extension files are played as raw files.

The audio mixer is initialized at the first occurrence of a \audio or \audio(…) tag.

Command **play**	Syntax: \audio(play=<filename>)
	Description: This command allows playing of a signal file at the specified position in the text. The filename can contain slash in order to specify a full path. Backslashes are not admitted, and you must use "%20" string for blanks, thus the syntax will be UNIX like, either in Windows. The <filename> can be an URL too (supported on Windows, on Linux by means of the library "libcurl.so" usually included in the Linux distributions, not supported on Solaris).

	Example 1:
	`This is \audio(play=music.wav) a test.`
	
	Result:
	"This is" will be pronounced, then music.wav will be played, then "a test" will be pronounced.
	
	Example 2:
	`This is \audio(play=music.wav;volume=50) a test.`
	
	Result:
	"This is" will be pronounced, then music.wav will be played at volume 50% (see volume command below), then "a test" will be pronounced.
	
	Example 3:
	`This is \audio(play=music1.wav;play=music2.wav) a test.`
	
	(equivalent)
	`This is \audio(play=music1.wav) \audio(play=music2.wav) a test.`
	
	Result:
	"This is" will be pronounced, then music1.wav will be played, then music2.wav will be played, finally "a test" will be pronounced.

Command **mix**	Syntax: \audio(mix=<filename>) *or* \audio(mix=<filename>,loop) *or* \audio(mix=<filename>,<count>)
	Description: This command allows playing of a signal file at the specified position in the text. The filename can contain slash in order to specify a full path. Backslashes are not admitted, and you must use "%20" string for blanks, thus the syntax will be UNIX like, either in Windows.

	Example 1:
	`This is \audio(mix=music.wav) a test.`
	
	Result:
	Speech and music.wav will be mixed together. The current track is music.wav (see the *track* command below for details).
	
	Example 2:
	`This is \audio(mix=music.wav,loop) a long test.`
	
	Result:
	Speech and music.wav will be mixed together. If the end of the audio file is reached, it will restart from the beginning. The current track is music.wav (see the *track* command below for details).
	
	Example 3:
	`This is \audio(mix=music.wav,3) a long test.`
	
	Result:
	Speech and music.wav will be mixed together. If the end of the audio file is reached, it will restart from the beginning 3 times. The current track is music.wav (see the *track* command below for details).
	
	Note:
	`\audio(mix=music.wav)` and `\audio(mix=music.wav,1)` are equivalent.

Command **name**	Syntax: \audio(name=<track name>)
	Description: This command allows setting a mnemonic name to the current track. This mnemonic name can be used in the **track** command instead of the file name (see below).

Command **volume**	Syntax: \audio(volume=<range(0-200)>)
	Description: This command allows setting the volume of the current audio track. To specify the current track use the *track* command (see below). Default volume is 100%. The range values are percentages of the default volume.

	Example 1:
	`This is \audio(mix=music.wav) \audio(volume=50) a test.`
	
	Result:
	The volume is set to 50% since the beginning.
	
	Example 2:
	`This is \audio(mix=music.wav) a test. Now I set The volume \audio(volume=50) to 50%.`
	
	Result:
	The volume is set to 50% after a while.

Command **pause**	Syntax: \audio(pause[=filename])
	Description: This command allows pausing the current audio track. To specify the current track use the *track* command (see below).

	Example 1:
	`\audio(mix=music.wav) Music mixing \audio(pause) is now in pause.`
	
	Result:
	The mixing is suspended before the words "is now in pause".
	
	Example 2:
	`\audio(mix=music1.wav;mix=music2.wav) Music mixing \audio(pause=music1.wav) is now in pause. The current track is now music1.wav.`

Command **resume**	Syntax: \audio(resume[=filename])
	Description: This command allows resuming the current audio track. To specify the current track use the *track* command (see below). If the track is not in pause (see pause command) it has no effect.

	Example 1:
	`\audio(mix=music.wav) Music mixing \audio(pause) is now in pause. \audio(resume) Mixing is working again.`
	
	Result:
	The mixing is suspended before the words “is now in pause”. Then it’s working again.
	
	Example 2:
	`\audio(mix=music1.wav;mix=music2.wav;mix=music3.wav) Music mixing \audio(pause=music1.wav;pause=music2.wav) is now in pause. \audio(resume=music2.wav) Mixing is working again.`
	`The current track is now music2.wav.`

Command **pauseall**	Syntax: \audio(pauseall)
	Description: This command allows pausing all the audio tracks. It is possible to resume audio tracks paused using the *resume* command or the *resumeall* command.

	Example:
	`\audio(mix=music1.wav) \audio(mix=music2.wav) This is a test using \audio(pauseall) the mixing feature.`
	
	(equivalent)
	
	`\audio(mix=music1.wav;mix=music2.wav) This is a test using \audio(pauseall) the mixing feature.`
	
	Result:
	The command will stop both the audio files.

Command **resumeall**	Syntax: \audio(resumeall)
	Description: This command allows resuming all the paused audio tracks.

	Example: `\audio(mix=music1.wav)\audio(mix=music2.wav) Music mixing \audio(pauseall) is now in pause. \audio(resumeall) Mixing is working again.` Result: The mixing is suspended before the words "is now in pause". Then it's working again.

Command **stop**	Syntax: \audio(stop[=filename])
	Description: This command allows stopping the last audio track. To specify the current track use the *track* command (see below). It is not possible to resume an audio track using the *resume* command, after a *stop* command.

	Example 1:
	`\audio(mix=music.wav) Music mixer \audio(stop) is now stopped.`
	
	Example 2:
	`\audio(mix=music1.wav;mix=music2.wav) This is a test. \audio(stop=music1.wav) music1 is now stopped.`

Command **stopall**	Syntax: \audio(stopall)
	Description: This command allows stopping all the audio tracks. It is not possible to resume an audio track using the *resume* command, after a *stopall* command.

	Example:
	`\audio(mix=music1.wav) \audio(mix=music2.wav) This is a test using \audio(stopall) the mixing feature.`
	
	(equivalent)
	
	`\audio(mix=music1.wav;mix=music2.wav) This is a test using \audio(stopall) the mixing feature.`
	
	Result:
	The command will stop both the audio files.

Command **path**	Syntax: \audio(path=<path>)
	Description: This command allows specifying a common path where the audio files are stored.

	Example:
	`\audio(path=c:/signals) \audio(mix=music1.wav) This is a test. \audio(mix=music2.wav) Hello world. \audio(path=c:/oldsignals) \audio(play=music3.wav) .`
	
	(equivalent)
	
	`\audio(path=c:/signals;mix=music1.wav) This is a test. \audio(mix=music2.wav) Hello world. \audio(path=c:/oldsignals;play=music3.wav) .`
	
	Result:
	The file music1.wav and music2.wav will be searched in the local folder c:\signals.
	The file music3.wav will be searched in the local folder c:\oldsignals.

Command **track**	Syntax: \audio(track=<filename.wav>)
	Description: This command allows specifying which track is considered as the current track.

	Example:
	`\audio(mix=music1.wav) The current track is music1.wav.`
	`\audio(mix=music2.wav) Now the current track is music2.wav.`
	`\audio(track=music1.wav;pause) The “pause” command is referred to the music1.wav track. Now the current track is music1.wav.`
	`\audio(track=music2.wav;volume=50) The volume of music2.wav is set to 50%. Now the current track is music2.wav`
	
	Note:
	If the current track ends or is stopped, a new current track would be selected from the active ones, using the *track* command.

Command **mix2play**	Syntax: \audio(mix2play[=filename])
	Description: This command switches the current track from mix mode to play mode. It is useful to complete the play of a file of unknown duration.

	Example 1:
	`\audio(mix=music.wav) The audio file is mixed with this sentence. \audio(mix2play) This sentence will be read after the end of music.wav`
	
	Example 2:
	`\audio(mix=music.wav,loop) The audio file is mixed with this sentence. \audio(mix2play) This sentence will be read after the end of music.wav. The 'loop' directive in the mixing command is ignored by mix2play.`

Command **fadein**	Syntax: \audio(fadein=<msec>)
	Description: This command allows setting a 'fade in' effect for the current track. To specify the current track use the *track* command.
	Example: `\audio(mix=music.wav) \audio(fadein=500) The audio file is mixed with this sentence and faded.`

Command **fadeout**	Syntax: \audio(fadeout=<msec>)
	Description: This command allows setting a 'fade out' effect for the current track. To specify the current track use the *track* command.
	Example: `\audio(mix=music.wav) The audio file is mixed with \audio(fadeout=500) this sentence and faded.`

Command **recstart/recstop**	Syntax: \audio(recstart=<track name>) \audio(recstop)
	Description: These commands allow recording speech that can be used in another part of the text.
	Example: `\audio(recstart=MyTrack1) Try this example using the recording capability. \audio(recstop;resume) 1234567890.` Result: The phrase and the numbers will be pronounced together.

Command **close**	Syntax: \audio(close)
	Description: This command allows closing the mixer. All the tracks are stopped and memory freed. Further \audio or \audio(…) tags will reinitialize the audio mixer.
	Example: `\audio(mix=music.wav) The audio file is mixed with this sentence. \audio(close) Mixer flushed. \audio Now the audio mixer is initialized.`

5.22 Bookmarks

\k<num>	Insert a bookmark. This tag inserts a bookmark in the text: when the text-to-speech engine encounters this tag, it notifies the application by calling the user callback and signaling that the bookmark has been reached. Note: this feature is implemented only with bookmark capable audio destinations (such as the Windows multimedia). It is generally used by user's applications to have a callback point.

6 Tools and Samples

6.1 Console applications

NOTE: The SAPI5 and SAPI4 samples apply only to Loquendo TTS for Windows.

These console applications are included along with their source code:

- HelloTTS_AudioBoard (reads a single Italian sentence)
- HelloTTS_RawFile (produces a RAW audio file containing a single Italian sentence)
- HelloTTS_WavFile (produces a Windows .WAV audio file containing a single Italian sentence)
- HelloTTS_SAPI5_AudioBoard (reads a single Italian sentence using Microsoft SAPI 5)
- HelloTTS_SAPI5_WavFile (produces a Windows .WAV audio file containing a single Italian sentence using Microsoft SAPI 5)
- HelloTTS_SAPI4_AudioBoard (reads a single Italian sentence using Microsoft SAPI 4)
- HelloTTS_SAPI4_WavFile (produces a Windows .WAV audio file containing a single Italian sentence using Microsoft SAPI 4)
- LoqActiveX_VBSample (Visual Basic sample using Loquendo ActiveX)
- LoquendoTTSFileGenerator (produces a set of audio files according to the specified parameters – a ReadMe.txt file is included in the distribution)

All these applications use the Italian Robotic male voice "Mario" (shipped with the Loquendo TTS SDK).

6.2 Web applications

NOTE: This section applies only to Loquendo TTS for Windows (unless differently specified).

These web applications are included:

- HelloTTS_HTML (HTML sample to test locally the Loquendo TTS ActiveX)
- HelloTTS_Server (ASP sample for client/server application)

By default, all these web pages use the Italian Robotic male voice "Mario" (shipped with the Loquendo TTS SDK).

6.3 Multi-platform GUI application

These multi-platform sample applications are shipped with Loquendo TTS SDK:

- TTSDirector

6.3.1 TTSDirector

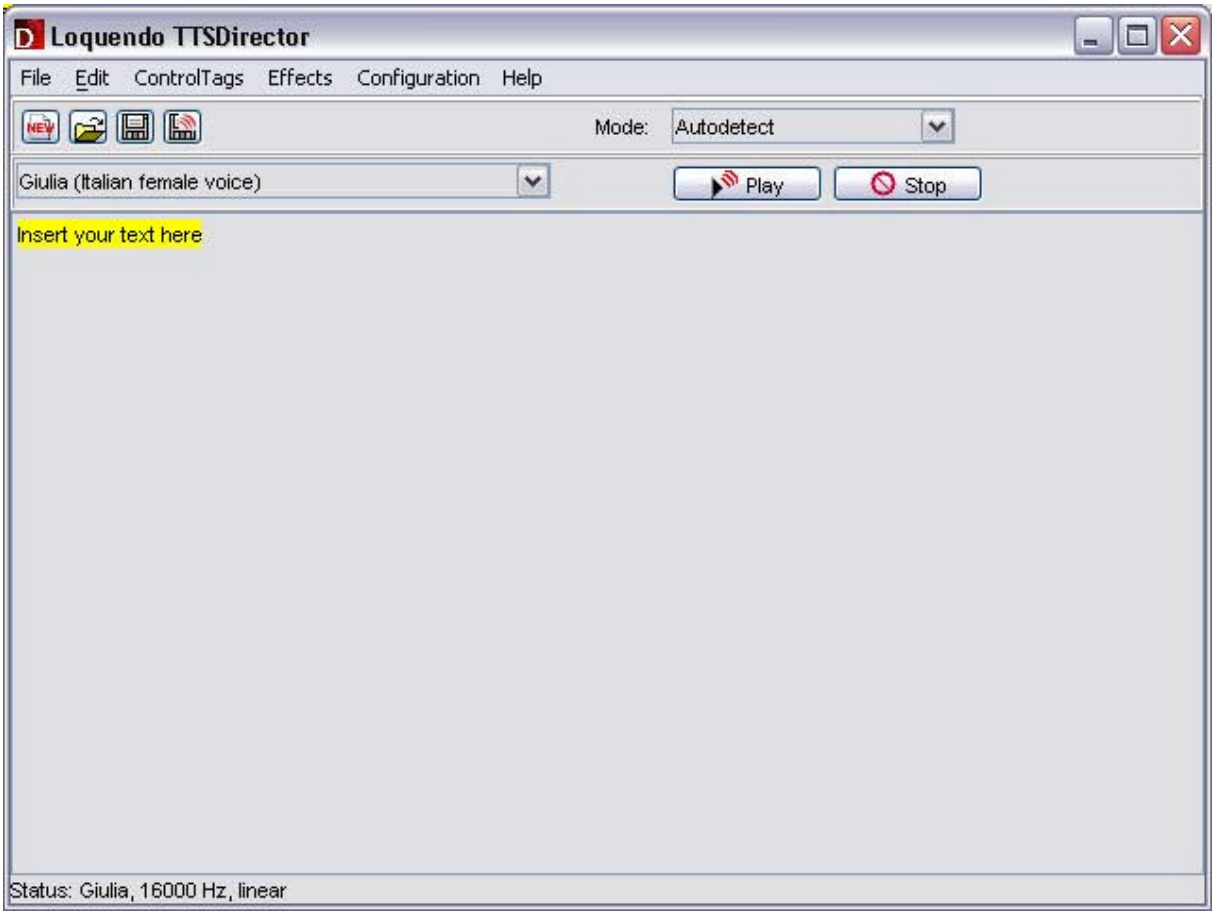
Loquendo TTS Director is a Java multi-platform development tool intended for helping the user in the design of his application prompts.

The text of the application prompt can be written in the edit box and interactively refined by means of a "listen & edit" procedure, allowing to tune the TTS behavior by means of the Loquendo TTS User Control Tags. A detailed menu helps choosing the proper tags. The tuned prompt can be saved as a text or as an audio file.

The allowed encodings for the input text are (Western European) ISO Latin 1, that is ISO-8859-1, and UNICODE UTF8 and UTF16.

TTSDirector needs the Java Runtime Environment (JRE) version 1.4.2 (at least), that it is installed during the SDK installation procedure (on request). In any case, you can find the 1.4.2 version of the JRE in the SDK CD-ROM distribution.

This is a screenshot of TTSDirector[4]:



[4] This application may be subject to minor changes to its interface – this screen shot may be different

Two combos allow selecting, respectively, the default **TTS voice** (that may be changed via control tags in the texts) and the **Mode** (Multi-line, Paragraph, SSML, see paragraph 2.1). In a similar way, font type and font dimension can be changed by means of other two combos.

The buttons **Play** and **Stop** allow synthesizing the edited text with Loquendo TTS.

The **File** menu allows opening and saving the edited prompts, both in text and audio formats.

The **Edit** menu allows Cut & Paste in the edit window (also available via left mouse button).

The **ControlTags** menu provides a structured access to the available Loquendo TTS Control Tags. The Tags are grouped according to their categories (see the Control Tags Paragraph in this Guide), so that it is easy to choose the intended one. The selected control is automatically inserted in the edit box, at the caret position (the "caret" is a flashing line, block, or bitmap in the client area of a window or in a control that accepts keyboard input). It indicates the place at which text or graphics are inserted. In case the control needs further specification by the user, this is marked by a yellow text in the edit box, asking for the needed details. E.g.:

\voice=<insert a valid voice name>

The **Effects** menu is a guide to the advanced features of "expressive cues" and "plugin lexicons". In case the selected voice is provided with such special add-ons, this menu allows selecting the desired effect.

The repertoire of *Expressive Cues* consists of a set of pre-recorded formulas, comprising conventional figures of speech, like greetings and exclamations ("hello!", "oh no!", 'I'm sorry!"), interjections ("Oh!", "Well!", "Hum"..) and paralinguistic events (e.g. breath, cough, laughter, etc.), which suggest expressive intention (to confirm, doubt, exclaim, thank, etc.). The use of such formulas can make vocal messages lifelike and expressive. The Effects menu allows selecting the proper formulas among those available for the active voice. The linguistic formulas are listed in the ***SpeechActs*** submenu, according to intuitive linguistic categories. The paralinguistic events are accessible from the ***Extras*** submenu. The selected expression is directly inserted in the edit box.

Every "SpeechAct" or "Extra" is played when the mouse pointer pass on the loudspeaker icon, in order to have a faster select of the proper Expressive Cue.

The ***Plugin*** submenu allows activating/deactivating the plugin lexicons available for the current voice. The selected plugin lexicon (see the relative paragraph in this Guide) is activated on the edited text from the caret position onward, until explicit de-activation.

The **Tools** menu allows activating, at the present time, the "Loquendo LexEditor" tool (see the paragraph 6.4.2 for more information about LexEditor), but only in the WINDOWS™ environment.

The **Configuration** menu allows setting some acoustic and prosodic parameters for the Loquendo TTS voices: sampling frequency and coding, pitch, speaking rate and volume.

More edit instances (panes with a tab) can be opened and saved in a single TTSDirector session, in order to build and test several voice prompts at the same time. The "New" button or the "CTRL-t" key can be used to switch between the instances. Separate Cut-Copy-Paste popup menus are available for every instance, and can be activated a click of the right button of the mouse in the editor area. A similar click of the right button on the editor's tab activate a Save-Save as-Close popup menu, and can be used to save the data present in the relative editor instance.

This is a short list of the available keys:

- "CTRL-t" : create a new editor instance
- "CTRL-tab" : go to the next editor instance
- "CTRL-Shift-Tab" : go to the previous editor instance
- "CTRL-z" : undo (that is, undo the last editing)
- "CTRL-y" : redo (that is, redo the last editing)

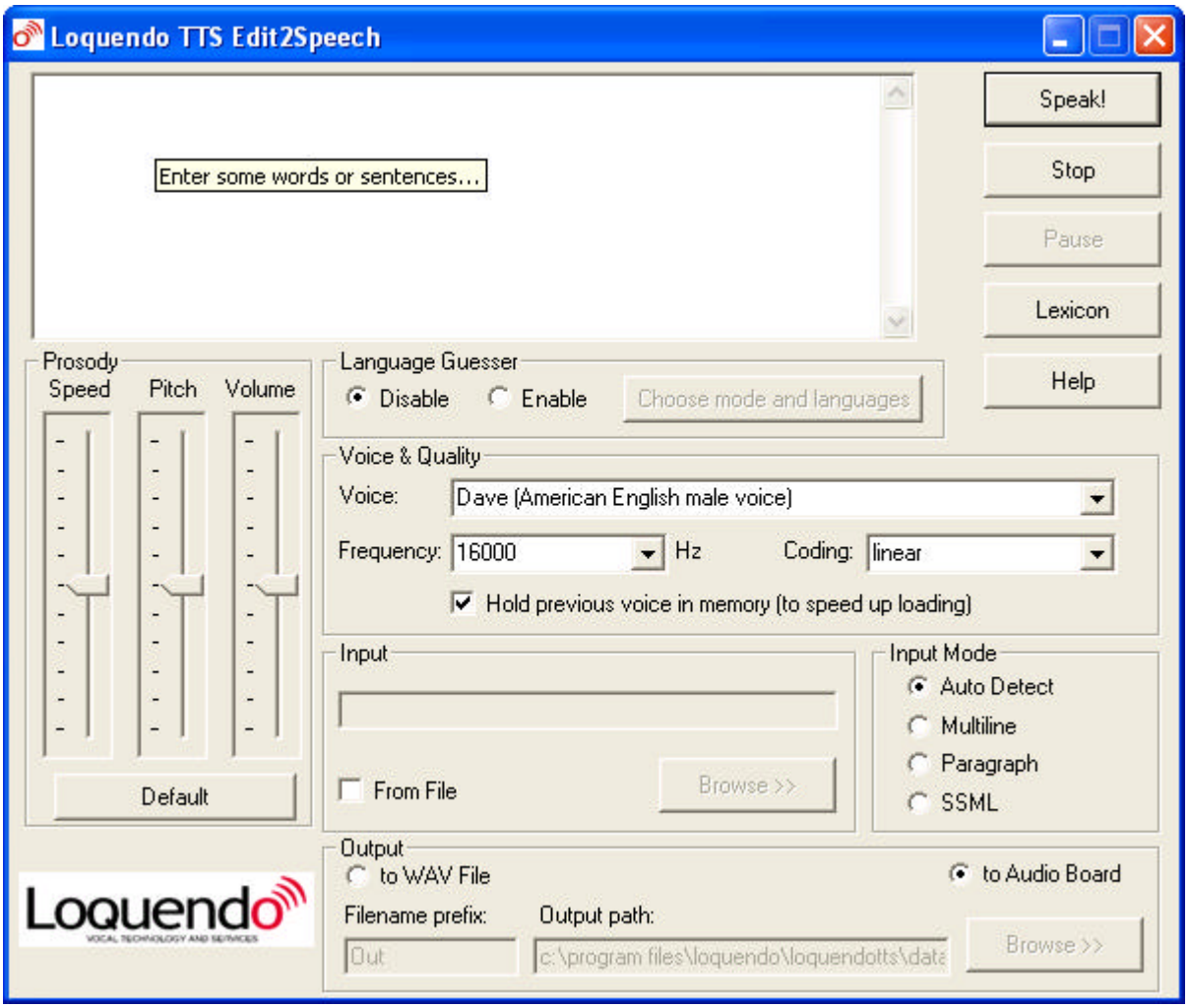
6.4 Windows only GUI application

These Windows sample applications are shipped with Loquendo TTS SDK:

- Edit2Speech
- LexEditor
- Eloqwi
- TTSApp
- TTSDirUpdate

6.4.1 Edit2Speech

This is a screenshot of Edit2Speech[5]:



[5] This application may be subject to minor changes to its interface – this screen shot may be different

This program reads the contents of its edit box, as soon as button "Speak!" is pressed. Stop and Pause/Resume buttons allow interactive speaking control. Three slides and a "Default" button control Speed, Pitch and Volume. There is the chance of reading input from a text file, instead of the edit box.

The sampling frequency and the signal coding (i.e. linear PCM, A-law PCM and μ-law PCM) can be selected too.

Even if one voice ha been selected, it's easy to switch from a voice to another, embedding a specific tag ("\voice=") in the text. For instance:

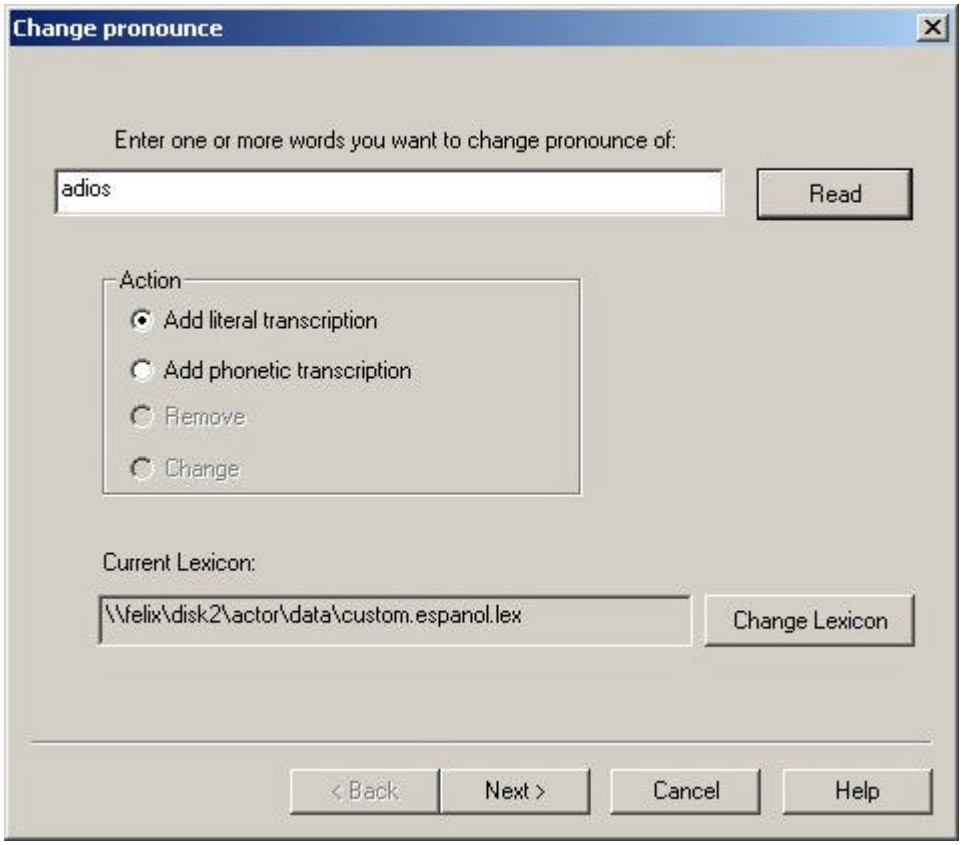
\voice=Susan Hello, my name is Susan. **\voice=Dave** Hi, Susan. My name is Dave. How are you?

The TTS output can be redirected to a WAV file, which is playable by any Windows file player. Each sentence is saved into a different file, whose name has a common prefix and a progressive number.

At the bottom of the main dialog, a radio button named "InputMode" allows changing of the Reading mode, from "Multiline", to "Paragraph", "SSML" or "Autodetect", that is the default one. See the Loquendo TTS User Guide for details.

It is possible to Enable/Disable the Language Guesser by means of two radio buttons, but in order to get the automatic language detection, you need to have installed the CD "Mixed Language Capabilities" (optional).

Pressing the Lexicon button and follow instructions to open a new dialog:



This dialog allows changing of words pronunciation. There are four options:

- Adding a literal transcription
- Add phonetic transcription

- Remove transcription
- Change transcription

Choosing the first one will open a second dialog where the user can enter a literal transcription for a word. The change will be immediately effective and will remain active until differently specified. The second option allows entering a custom phonetic transcription (the phoneme symbols used are described in the Loquendo TTS User Manual).

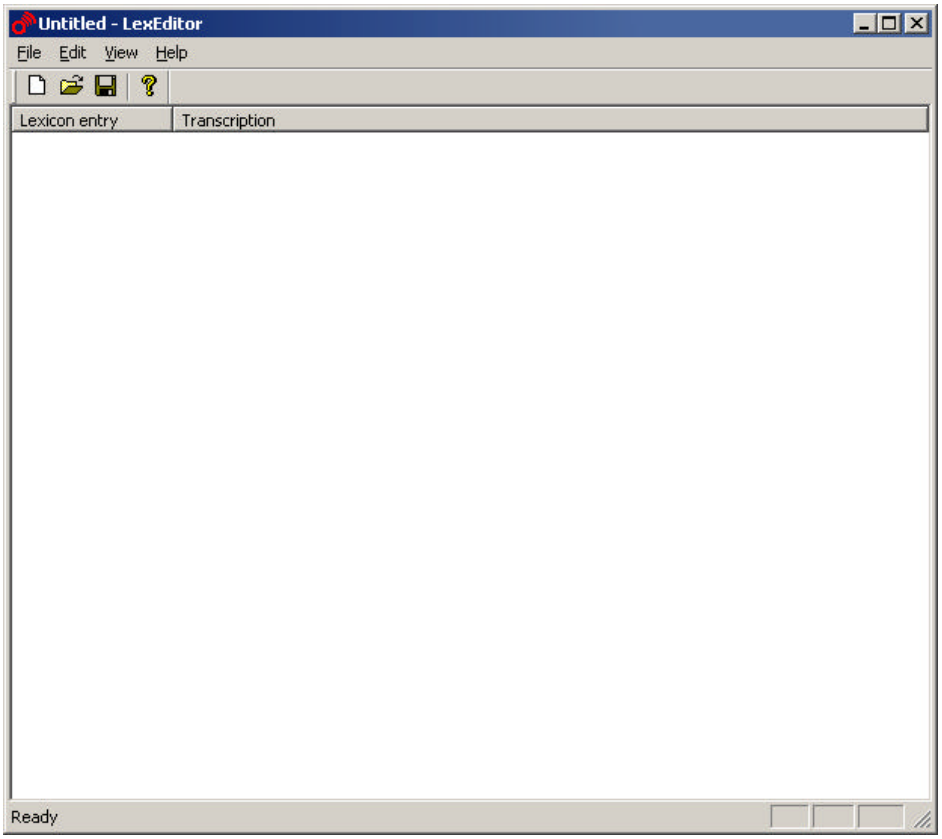
If a literal or phonetic transcription is already present in the Loquendo TTS lexicon, it can be removed or changed.

Even the position of the Loquendo TTS lexicon file may be changed from here.

6.4.2 LexEditor

This application allows creating and editing user lexicon files. It can be used as a stand alone program, to be run with "LexEditor.exe", or can be activated by means of the **Tools** menu of the TTSDirector application (see paragraph 6.3.1), but only in the WINDOWS™ environment.

Running `LexEditor.exe`, the following window is shown:

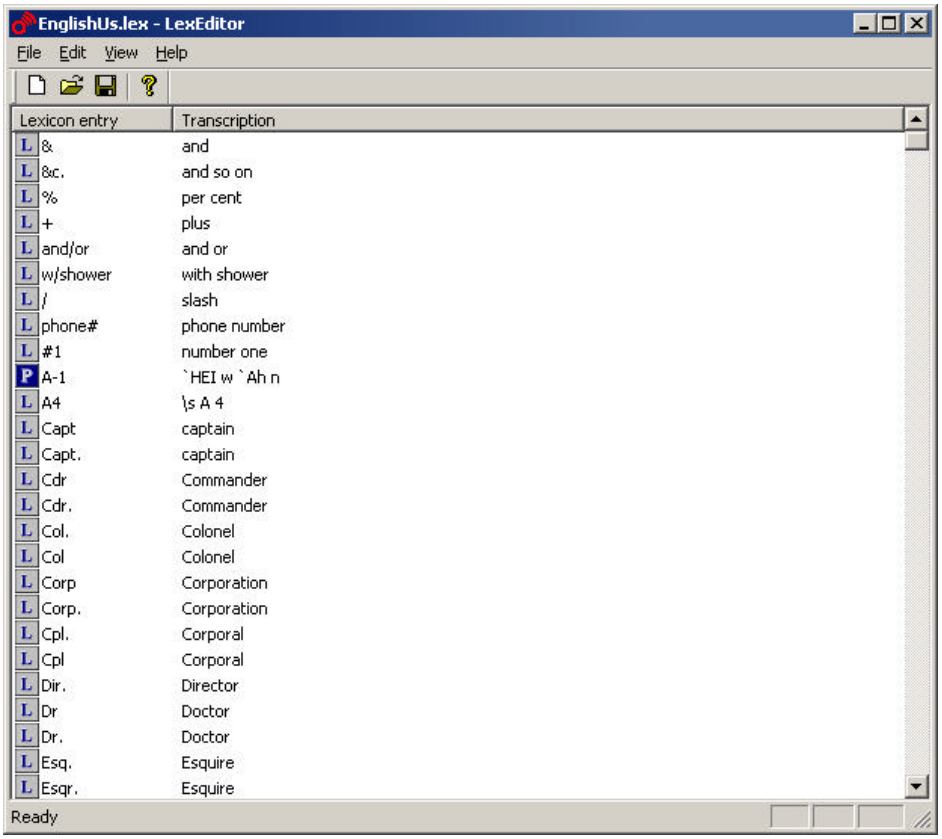


The application menu provides the following functionalities:

- **File → New** (also through the *Ctrl-N* shortcut or the button in the toolbar): creates a new lexicon file;
- **File → Open** (also through the *Ctrl-O* shortcut or the button in the toolbar): opens an existing lexicon file;
- **File → Save** (also through the *Ctrl-S* shortcut or the button in the toolbar): saves the current lexicon file;
- **File → Save As**: saves the current lexicon file with a different name;
- **File → 1, … File → 4**: opens the last recently used lexicon files, if any;
- **File → Exit**: exits the application;

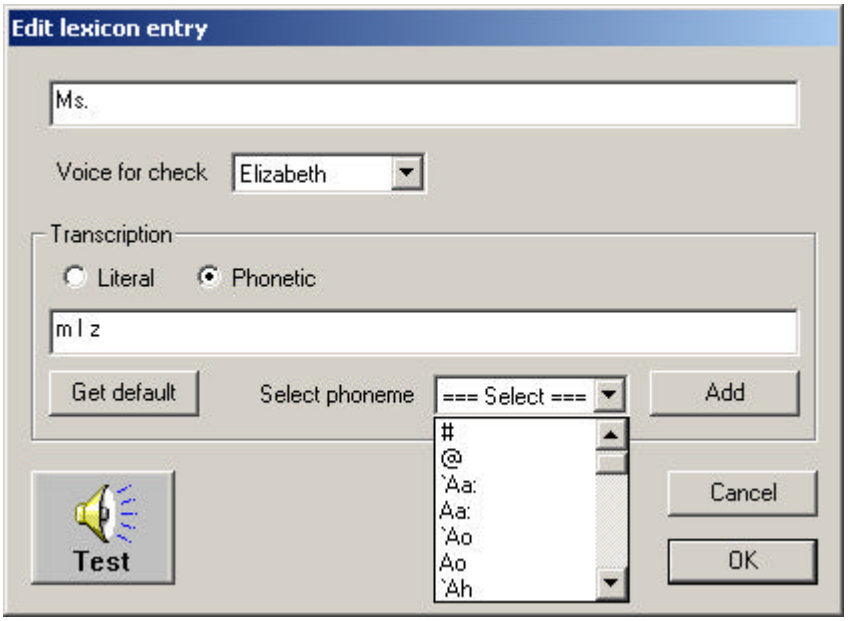
- **Edit → Insert** (also through the *Ctrl-I* shortcut): shows the lexicon dialog (see below) to insert a new entry in the current file; confirming the dialog, the new lexicon entry will be inserted before the currently selected entry in the editor;
- **Edit → Delete** (also through the *DEL* shortcut): deletes, upon notice, the currently selected lexicon entry in the editor;
- **Edit → Import list** (also through the *Ctrl-M* shortcut): opens a text file and shows the import dialog (see below) to insert the default transcriptions of selected words at the end of the current lexicon file;
- **View → Toolbar** (toggle): hides/shows the toolbar;
- **File → Status Bar** (toggle): hides/shows the status bar at the bottom;
- **Help → About** (also through the button in the toolbar): shows version information for the LexEditor.

When opening an existing lexicon file, the contents of the file are listed in the editor as follows:



The L and the P icons stand for literal transcription or phonetic transcription, respectively.

Double-clicking a lexicon entry in the list, you can edit it through the lexicon dialog:



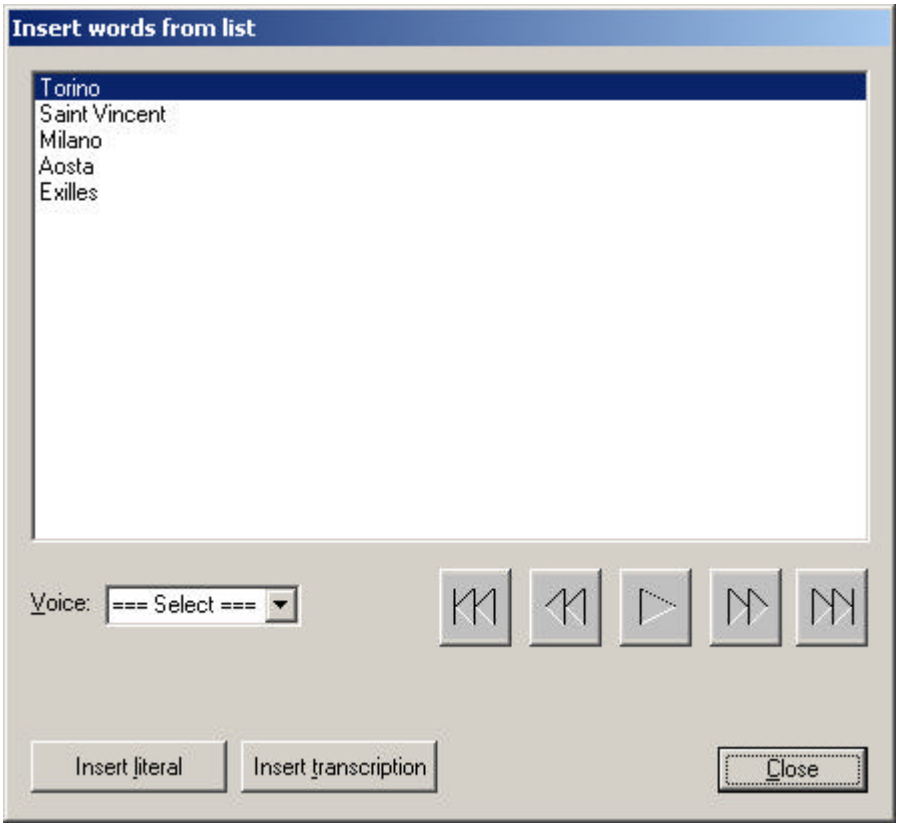
Selecting a Loquendo TTS voice in the *Voice for check* list, you can:

- have a feedback about the correctness of the phonetic transcription: the text in the transcription edit box turns to red when it contains characters not allowed for the language of the selected voice;
- get the default phonetic transcription for the lexicon entry, by pressing the *Get default* button;
- get the list of the existing phonemes[6] for the language of the selected voice and insert them in the new transcription by pressing the *Add* button;
- hear the sound of the new transcription, by pressing the *Test* button.

The same lexicon dialog appears when you want to add a new lexicon entry in your file using the *Edit* → *Insert* menu item.

Finally, by means of the *Edit* → *Import list* option you can build up a lexicon starting from an existing list of words (a text file, one word per line). By listening to the words sequentially synthesized, you can select those needing some re-adjustment. The selected words will be inserted in a lexicon together with their default transcription, that you can subsequently modify by double clicking on each item (see above). If you use the *Edit* → *Import list* menu item, after asking for the pathname of the text file you want to import, the following dialog box will appear:

[6] The phonemes are shown using the Loquendo syntax described in the language specific reference manuals



Selecting a Loquendo TTS voice in the *Voice* list, you can:

- hear the sound of the selected word or the next, previous, first or last one, by pressing the corresponding button;
- insert at the end of the current lexicon file the default literal or phonetic transcription of the selected word (to edit later on), by pressing the *Insert literal* or the *Insert transcription* button.

6.4.3 Eloqwi

This is a Windows clipboard reader. This application looks like a small red mouth in the system tray:

Eloqwi can be used in conjunction with any text editor or word processor, for easily navigating inside a long or complex document. To access its additional functionalities (such as voice changing), point the small red mouth and click the right mouse button.

6.4.4 TTSApp

TTSApp is a Microsoft re-distributable application that allows testing of a SAPI engine. The application search the computer for any SAPI 5 compliant engines, and interacts with them, calling some of the "required" SAPI interfaces. Running TTSApp is probably the simplest method to know whether SAPI TTS engines have been correctly installed. Further information on TTSApp can be found in the Microsoft SAPI 5 documentation.

6.4.5 AttsTest

AttsTest is a Microsoft re-distributable application that allows testing of a SAPI engine. The application search the computer for any SAPI 4 compliant engines, and interacts with them, calling some of the "required" SAPI interfaces. Running AttsTest is probably the simplest method to know whether SAPI TTS engines have been correctly installed. Further information on AttsTest can be found in the Microsoft SAPI 4 documentation.

6.4.6 TTSDirUpdate

TTSDirUpdate is a simple application that should be run whenever one or more Loquendo TTS voices have been installed or moved, in order to save the new configuration inside the Windows registry.

7 APPENDIX A: XML support

Loquendo™ TTS supports Voice XML 1.0 and Voice XML 2.0, assuming that its reading mode has been setup as “xml” or “wxml” (input text in Unicode code format) or “w8xml” (input text in UTF-8 code format), by using the appropriate API (*ttsSetReadingMode)* described in the Loquendo™ TTS Programmer’s Guide.

The voice XML 1.0 variant will be recognized by means of the first-level tag <PROMPT>, the voice XML 2.0 whit first-level tag <SPEAK>.

The three **<pros>** and **<prosody>** attributes can be specified as follows:

mode	*meaning*
n	specifies the attribute value (e.g. rate=”110” , 110 words per minute)
+n	Increase by n the attribute value (e.g. pitch = +15, increase pitch by 15 hz)
-n	Decrease by n the attribute value (e.g. pitch = +15, decrease pitch by 15 hz)
+n%	Increase the attribute value by n percent (e.g. vol = “+30%”)
-n%	Decrease the attribute value by n percent (e.g. vol = “-30%”)
reset	Resets the attribute value (to default)

7.1 VOICEXML 1.0: SUPPORTED TAGS AND FORMATS

TAGS			SUPPORT	FORMATS	EXAMPLES
Break	Msecs		supported	Standard	This <break msecs="5000"/> is a 5 seconds pause.
	size (none, small, medium, large)		supported	Standard	This <break size="large"/>is a long pause.
Div	type	Sentence	supported	Standard	my sentence
		Paragraph	supported	Standard	my paragraph
Emp	level (strong, moderate, none, reduced)		supported	Standard	Today is a <emp level="strong"> very</emp> important day.
Pros	rate [7]		supported	Standard	<pros rate="-20%"> Slow pitch sentence </pros>
	Vol		supported	Standard	<pros vol="+20"> High pitch sentence </pros>
	Pitch		supported	Standard	<pros pitch="+10%"> High pitch sentence </pros>
	Range		Not supported		

[7] The possible formats are reassumed in the previous table.

TAGS			SUPPORT	FORMATS	EXAMPLES
Sayas	phon		Not supported		
	sub		supported	standard	<sayas sub="hi"> hello </sayas>
	class	phone	supported	standard	<sayas class="phone"> 349 4640690 </sayas>
		date	supported	standard	Standard: <sayas class="date"> 12/12/2000 </sayas>
		digits	supported	standard	<sayas class="digits"> 12345 </sayas>
		literal	supported	standard	<sayas class="literal"> 12345 </sayas>
		currency	Not supported		
		number	supported	standard	<sayas class="number"> 12345 </sayas>
		time	supported	standard	<sayas class="time"> 23:12:23 </sayas>

7.2 SSML 1.0 (W3C WD 02 December 2002): SUPPORTED ELEMENTS AND FORMATS

ELEMENTS AND ATTRIBUTES	SUPPORT	NOTE	EXAMPLES
speak	supported	required	<speak version="1.0"> 123. </speak>
version (speak attribute)	supported	required	
xml:lang (attribute)	supported	required	<speak version="1.0" xml:lang="en"> 123. </speak>
xml:base (speak attribute)	not supported		
xmlns (speak attribute)	not supported		
xmlns:xsi (speak attribute)	not supported		
xsi:schemaLocation (speak attribute)	not supported		
lexicon	supported	Absolute path + filename URI format: file://..... May occur as immediate children of the speak element	<speak version="1.0" xml:lang="en"> <lexicon uri="file://mypcname/lexicon.lex"/>Hello. </speak>
meta	supported	not used	

name (meta attribute)				supported	cross control with “http-equiv”	
http-equiv (meta attribute)				supported	cross control with name	
content (meta attribute)				supported	required	
matadata				supported	not used	

p				supported		`<speak version=”1.0” xml:lang=”en”>`
						`<p> my paragraph</p>`
						`</speak>`

xml:lang (attribute)				supported		`<speak version=”1.0” xml:lang=”it”>`
						`123`
						`<p xml:lang=”en”> my paragraph</p>`
						`</speak>`

s				supported		`<speak version=”1.0” xml:lang=”en”>`
						`<s> my sentence </s>`
						`</speak>`

xml:lang (attribute)				supported		`<speak version=”1.0” xml:lang=”it”>`
						`123`
						`<s xml:lang=”en”> my sentence </s>`
						`</speak>`

say-as	**interpret-as**	**format**	**detail**			

	letters			supported		`<speak version="1.0" xml:lang="en">`
						`<say-as interpret-as="letters"> USA </say-as>`
						`</speak>`

	words			supported		`<speak version="1.0" xml:lang="en">`
						`<say-as interpret-as="words"> USA </say-as>`
						`</speak>`

	number			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="number"> 234512 </say-as> </speak>
	number	cardinal		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="number" format="cardinal"> 234512 </say-as> </speak>
	number	ordinal		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="number" format="ordinal"> VIII </say-as> </speak>
	number	telephone		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="number" format=”telephone”> 347 2324769</say-as> </speak>
	number	digits		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="number" format="digits"> 234512 </say-as> </speak>
	date	mdy, ymd, ym, my, md, y, m, d		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="date" format="ymd"> 2002/12/02 </say-as> </speak>
	time			supported	hh:mm:ss hh:mm	<speak version="1.0" xml:lang="en"> <say-as interpret-as="time"> 23:05:16 </say-as> </speak>
	currency			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="currency">13,23$ </say-as> </speak>
	measure			not supported		

	telephone			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="telephone"> 347 2324769</say-as> </speak>
	name			not supported		
	net	email		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="net" format="email"> name.surname@loquendo.com</say-as> </speak>
		uri		supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="net" format="uri"> http://www.loquendo.com</say-as> </speak>
	vxml:boolean			not supported		
	vxml:date			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:date">19630510</say-as>. </speak>
	vxml:digits			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:digits"> 123456 </say-as> </speak>
	vxml:currency[8]			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:currency"> eur10.32</say-as> </speak>

[8]

Language	Character Currency Indicator
Italian	EUR, USD, GPB, JPY
French	EUR, USD, GPB, JPY
German	EUR, USD, GPB, JPY
Spanish (and sublanguage: Es:Mexican)	EUR, USD, GPB, JPY,ESP
English (and sublanguage ES:American)	EUR, USD, GPB, JPY

Only these languages accept currency indicator.

	vxml:number			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:number"> 123454</say-as> </speak>
	vxml:phone			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:phone">+39 333 866592</say-as> </speak>
	vxml:time			supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="vxml:time">0921pm</say-as> </speak>
	address			not supported		
			dictate	supported		<speak version="1.0" xml:lang="en"> <say-as interpret-as="" detail="dictate"> It's simple, isn't it?</say-as> </speak>

ELEMENTS AND ATTRIBUTES		SUPPORT	NOTE	EXAMPLES
phoneme		supported		`<speak version="1.0" xml:lang="en">` `<phoneme ph="T$-Ae-`Oa:">hello</phoneme>` `</speak>`
ph (phoneme attribute)		supported	required	

alphabeth (phoneme attribute)		supported	optional
			
			Loquendo TTS's phonemes (default)
				`<speak version="1.0" xml:lang="en">`
				`<phoneme alphabet="x-loquendo" ph="T$-Ae-`
				`Oa:">hello</phoneme>`
				`</speak>`

			IPA phonemes [9]	`<speak version="1.0" xml:lang="en">`
				`<phoneme alphabet="ipa"`
				`ph="ʧæɔˈː">hello</phoneme>`
				`</speak>`

sub		supported		`<speak version="1.0" xml:lang="en">`
				`<sub alias="World Wide Web Consortium">W3C</sub>`
				`</speak>`

voice	xml:lang	supported		

	gender	supported		`<speak version="1.0" xml:lang="en">`
				`<voice gender="female">This is a female voice.</voice>`
				`</speak>`

	age	supported		

	variant	supported		`<speak version="1.0" xml:lang="en">`
				`<voice gender="female" variant="2"> This is another female voice.</voice>`[10]
				`</speak>`

[9] Use a space as separator between the phonetic transcription of different words.
[10] Variant is the sequence number of the preloaded Voices. Es:if the squence of the preloaded voices is: Sonia, Mario, Valentina, Silvana, Roberto, the female variant 2 is Valentina.

	name[11]	supported		<speak version=”1.0” xml:lang=”en”> <voice name=”Dave”>This sentence is read by Dave.</voice> </speak>
emphasis	level	supported		<speak version=”1.0” xml:lang=”en”> Today is a <emphasis level=”strong”>very</emphasis> important day. </speak>
break	strength	supported		<speak version=”1.0” xml:lang=”en”> Break test <break strength="strong"/> Goodbye. </speak>
	time	supported		<speak version=”1.0” xml:lang=”en”> This <break time=”4s”/> is a very long pause. </speak>
prosody	pitch	supported	standard + absolute variation (Hz) + percentual variation	<speak version=”1.0” xml:lang=”en”> <prosody pitch="high"> High pitch sentence </prosody> </speak>
				<speak version=”1.0” xml:lang=”en”> <prosody pitch="+20"> High pitch sentence </prosody> </speak>
				<speak version=”1.0” xml:lang=”en”> <prosody pitch="+60%"> High pitch sentence</prosody> </speak>
	contour	supported		<speak version=”1.0” xml:lang=”en”> <prosody contour="(0%,+20Hz)(10%,+30%)(40%,+10Hz)">good morning</prosody> </speak>
	range	supported		<speak version=”1.0” xml:lang=”en”> <prosody range="x-high">good morning</prosody> </speak>
	rate	supported	standard + percentual variation	<speak version=”1.0” xml:lang=”en”> <prosody rate ="fast"> Fast rate sentence </prosody> </speak>
				<speak version=”1.0” xml:lang=”en”> <prosody rate ="230"> Fast rate sentence </prosody> </speak>

IMPORTANT:Do not mix prosody tags and voice switch tags, the result could be unforeseeable. The XML parser causes errors when the voice has not been loaded.[11]

				<speak version=”1.0” xml:lang=”en”> <prosody rate="-80.5%"> Slow rate sentence</prosody> </speak>
	duration	supported		<speak version=”1.0” xml:lang=”en”> <prosody duration="3s">good morning</prosody> </speak>
	volume	supported	standard	<speak version=”1.0” xml:lang=”en”> <prosody volume="loud">High volume sentence </prosody> <prosody volume="60.0">High volume sentence </prosody> </speak>
			absolute variation	<speak version=”1.0” xml:lang=”en”> <prosody volume ="+10"> High volume sentence </prosody> </speak>
			percentual variation	<speak version=”1.0” xml:lang=”en”> <prosody volume ="-40.4%"> High volume sentence </prosody> </speak>
audio[12]		supported	Absolute path + filename URI format: file://.....	<speak version=”1.0” xml:lang=”en”> <audio src="file://localhost/welcome.wav">Hello</audio> </speak>
mark		supported		<speak version=”1.0” xml:lang=”en”> Go from <mark name="here"/> here, to <mark name=”there”/> there! </speak>

[12] The *audio* supports 16 bit sound files, mono and stereo, with arbitrary sample rate frequency.
“. wav” files are supported and played.
“.mp3”, “.wma”, “.asf”, “.ogg”, “.avi”, “.mpg” are not supported and are not played.
“. raw” , “.pcm” and any other extension files are played as raw files.

desc	supported	LoquendoTTS not use text-only output mode	

Note: it's advise using control tags inside ssml formatted text against, especially if the equivalent ssml element exist.